

Modelando e Implementando Bancos de Dados Relacionais
Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 1

Introdução a Banco de Dados

Meta

Apresentar conceitos importantes e as principais características de um banco de dados.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Distinguir dados de informação;
2. Definir o que é Banco de Dados e Banco de Dados Relacional;
3. Descrever a Tecnologia de Banco de Dados;
4. Identificar as três fases de um projeto de Banco de Dados.

Nos tempos mais remotos...

Desde os primórdios, o homem gera e transmite informações. Primeiro, foram as pinturas rupestres, depois esculpiram a pedra e, mais tarde, chegaram ao **papiro**.

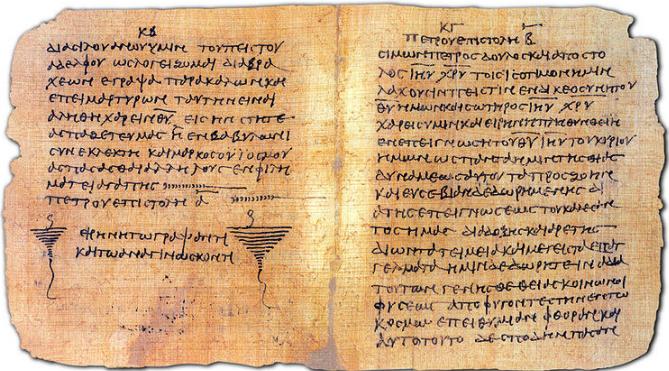


Figura 1.1 Pintura rupestre e papiro egípcio

Pintura Rupestre: Fonte: Flickr | Foto: Chico Ferreira

<http://www.flickr.com/photos/franciscoferreira/3981932151/>

Papiro: Fonte: Wikimedia Foundation | Foto: imagem em domínio público

http://commons.wikimedia.org/wiki/File:Papyrus_Bodmer_VIII.jpg

Inicio do verbete

Papiro

Obtido através da parte interna, branca e esponjosa do caule da planta do papiro. O caule era cortado em finas tiras que posteriormente eram molhadas, sobrepostas e cruzadas, para depois serem prensadas. A folha obtida era martelada, alisada e colada ao lado de outras folhas para formar uma longa fita que era depois enrolada. A escrita dava-se paralelamente às fibras. O papiro pronto era, então, enrolado a uma vareta de madeira ou marfim para criar o rolo que seria usado na escrita.

Fim do verbete

Com a invenção do papel, a informação ficou mais leve e, com o passar dos anos, houve a necessidade de transmitir conhecimentos e descobertas que cresciam a cada dia. O número de informações aumentou, assim como aumentou o número de pessoas que queriam se informar. Assim, as informações viraram livros e os livros lotaram as bibliotecas.



Figura 1.2. Com o aumento da disponibilidade de informações, o número de pessoas que desejavam se informar também aumentou.

Livro: Fonte: StockXchange | Foto: Ove Tøpfer

<http://www.sxc.hu/photo/993325>

Estantes com livros: Fonte: Flickr | Foto: Stewart and Vickie Carrington

<http://www.flickr.com/photos/stewickie/196586033/>

Para resolver o problema de espaço, isto é, as lotações das bibliotecas, criou-se o **microfilme**, mas este não tinha praticidade quanto à utilização, pois a principal desvantagem era a visualização, ou seja, a imagem era pequena demais para ser lida a olho nu. As bibliotecas usavam leitores especiais que projetavam as imagens maiores em telas de vidro horizontais.

[Inicio de Verbete](#)

Micro o quê?

É uma mídia analógica de armazenamento para livros, periódicos, documentos e desenhos. A sua forma mais padronizada é um rolo de filme fotográfico de 35mm preto e branco. O microfilme foi criado pelo francês Renée Dragon, no século XIX. Este foi utilizado pela primeira vez durante a guerra franco-prussiana, na qual pombos-correio transportavam um conjunto de dados em mapas micro filmados das posições inimigas.

[Fim do verbete](#)

O conhecimento gerava cada vez mais conhecimento que, como consequência, gerava mais informação. Desta maneira, mais pessoas procuravam informações ao mesmo tempo. A invenção do computador veio para buscar solucionar este problema do armazenamento e acesso à grande quantidade de informações que não paravam de progredir. Com o tempo, o computador foi se tornando uma tecnologia acessível a um número cada vez maior de pessoas. Mais computadores em uso geravam mais informações. A partir da necessidade de trocar as informações de forma mais rápida e eficiente, nasceu a internet.

[Início Box Multimídia](#)

Navegando pela história...

No site a seguir há informações bem interessantes sobre a invenção e desenvolvimento do computador. Vale a pena dar uma olhada, ou melhor, uma navegada. Acesse:

<http://www.discoverybrasil.com/internet/a-invencao-do-primeiro-computador.shtml>

Fim Box Multimídia



Figura 1.3. A internet possibilitou que pudéssemos ter acesso às informações de qualquer lugar do mundo em tempo real, conectando o maior número de pessoas possíveis.

Fonte: StockXchange | Foto: Sachin Ghodke

<http://www.sxc.hu/photo/1287371>

A Web aumentou ainda mais o volume e acesso às informações, crescendo exponencialmente – e sem cessar - até nossos dias, lotando os Bancos de Dados.

Hoje, os Bancos de Dados estão presentes em todos os lugares da indústria da tecnologia da informação (TI) e nos negócios em geral. Usamos os Bancos de Dados de maneira direta e indireta todos os dias – transações bancárias, reservas de viagem, relações de emprego, buscas em sites da Web, compras, vendas, etc. Estamos falando de um grande volume de dados, ou seja, informações que precisam ser armazenadas, acessadas e atualizadas por um indefinido espaço e tempo, proporcionando aos seus usuários agilidade e qualidade de resposta no cruzamento das informações.

E esta aula trata justamente sobre conceitos e características importantes de Banco de Dados. Vamos nessa?!

Fim da Introdução

Dados X Informações

Até aqui, você já ouviu falar de dados e de informações. Mas você sabe o que eles significam? São nomes diferentes para a mesma coisa? Definir esses conceitos é um passo necessário para compreender o que pode ser armazenado nos Bancos de Dados.

Dado: é qualquer elemento identificado em sua forma bruta que por si só, não conduz a uma compreensão de determinado fato ou situação. O dado pode ser apresentado na forma de números, palavras, imagens ou sons.

Ex: 39ºC.

Este dado isolado não diz nada. Ele relata apenas uma medida de temperatura em graus Celsius que pode estar associada a muitos eventos (temperatura corporal, temperatura do ambiente, etc.).

Informação: é o significado (semântica) do dado, isto é, o dado somado ao atributo, relevância e contexto. São dados coletados, organizados e ordenados.

$$\text{Informação} = \text{dado} + \text{significado}$$

Início Box de Curiosidade

O que cerveja tem a ver com fraldas?

Uma das maiores redes de varejo dos Estados Unidos, o Wal-Mart, descobriu em seu gigantesco armazém de dados (códigos de produtos vendidos), que a venda de fraldas descartáveis estava associada à de cervejas (informação). O que isso significa: que as crianças estão bebendo cerveja? Não! Uma investigação mais detalhada revelou que, em geral, eram os homens que saíam à noite para comprar fraldas e aproveitavam para levar algumas latinhas de cerveja para casa.

O setor responsável pela organização dos produtos não pensou duas vezes: os colocou lado a lado. Resultado: a venda de fraldas e cerveja disparou.

Então, temos que: dados = códigos de produtos vendidos e informação = associação

dos produtos fraldas e cerveja.

Fim box de Curiosidade

Já pensou tomar uma decisão importante com pouca informação? O risco de algo dar errado é grande, não é mesmo? Na tomada de decisões um Banco de Dados é uma fonte da qual podemos extrair uma grande variedade de informações.

Pois bem, como as informações são extraídas do Banco de Dados, não é possível processar informações em computadores, por exemplo. O que se processa nos computadores são os dados que representam essas informações. Só o ser humano é capaz de elaborar informações por meio de associações de conceitos. Assim, é correto falar em Banco de Dados ou Base de Dados e não em Banco de Informações.

Box de curiosidade

Momento de reflexão...

Antes de prosseguir com a leitura da aula, pare e reflita sobre a seguinte frase:

“Um indivíduo sem informações não pode assumir responsabilidades; um indivíduo que recebeu informações não pode deixar de assumir responsabilidades.”

Jan Carlzon, ex-presidente da Scandinavian Airlines System (SAS).

Fim do Box de curiosidade

Início da atividade

Atividade 1 - Atende ao objetivo 1

Dos textos abaixo, identifique o que se refere a dado e o que se refere à informação:

- a) As características de um indivíduo como, o peso de 80 kg e sua idade de 10 anos, revelam a natureza de uma criança obesa. Este fato levou um médico a propor um programa de alimentação adequada.
- b) Uma rede varejista descobriu que a venda de colírios aumentava na véspera dos feriados. (Por quê? Mistério...) Passou a preparar seus estoques e promoções do produto com base nesse cenário.

c) Uma empresa de telefonia detectou, ao implantar seu armazém de dados, que quatro grandes clientes empresariais eram responsáveis por mais da metade das chamadas de manutenção. Um deles estava prestes a abandonar os serviços. A telefônica fez reparos imediatos, convenceu o cliente a ficar e manteve uma receita anual de 150 milhões de dólares.

| | Dado | Informação |
|----|-------------|-------------------|
| a) | | |
| b) | | |
| c) | | |

Resposta Comentada

Como primeiro passo é importante distinguir dados e informação, pois o que é armazenado num Banco de dados são os dados e não informação. Se você não tiver isto em mente pode chegar a conceitos errados, tipo: “os seres humanos são máquinas e os computadores processam informações”. O computador processa só dado e nós, os seres humanos, processamos informações. Um dado pode ser informação para o receptor, mas do ponto de vista do computador, essa informação converte-se a um dado após ser armazenada. Com essas definições claras e objetivas você será capaz de reconhecer o que é um dado e o que é uma informação em qualquer situação ou caso real.

No item a, os dados são: 80 kg e 10 anos e a associação desses dados origina a informação, isto é, a criança obesa. Como resultado dessa informação, vemos o médico propondo um programa de alimentação adequada.

No item b, não está explícito o que é dado e o que é informação. O que podemos depreender é que os dados daquela rede varejista são códigos dos produtos vendidos, nomes dos produtos vendidos e a data de compra dos produtos, entre outros. Informação é o resultado da associação dos nomes dos produtos vendidos e a data de

compra, isto é, das vendas do colírio na véspera de feriados.

No item c, podemos considerar dados: nome do cliente, cargo, tipo de empresa, tipo e registro de chamadas, insatisfação do cliente, entre outros. Com base nesses dados, foi possível descobrir as seguintes informações: que quatro grandes clientes empresariais eram responsáveis por mais da metade das chamadas de manutenção e que um deles estava prestes a abandonar os serviços.

Fim da atividade

O que é Banco de Dados?

Uma imagem vale mais que mil palavras... ou não?



Figura 1.4. Banco de Dados. ([ilustração: fazer uma imagem baseada nesta ideia](#))

Sim, isso mesmo! Um grupo de dados juntos num só lugar, ou seja, no banco. Vamos relacionar os dados do jogo com os dados que são armazenados no computador, e o banco com o recipiente no qual armazena-se os dados, isto é, o Banco de Dados.

Mas é claro que esta é uma analogia bem geral do que seria um Banco de dados. A seguir, você estudará outras definições que estabeleçam um melhor entendimento formal do termo Banco de Dados:

Um Banco de Dados é uma coleção de arquivos (conhecido também como tabelas), que armazenam dados e suas respectivas associações.

Em um computador, os dados são organizados logicamente numa hierarquia que envolve campos, registros, arquivos e Bancos de Dados, tal como a escrita pode ser

organizada em textos, palavras, sentenças, parágrafos e documentos. A figura 1. 5 ilustra essa hierarquia de dados:

| | | | |
|----------------|---|-------------------------|--|
| Banco de Dados | Banco de dados vendas | | |
| | Arquivo cliente | Arquivo pedidos | Arquivo produtos |
| Arquivo | Arquivo cliente | | |
| | NOME Carla Soares Roberto L. João P. | IDADE 42 54 35 | ENDEREÇO Rua Meir, 33 Rua Beira, 11 Rua G, 67 |
| Registro | NOME Carla Soares | IDADE 42 | ENDEREÇO Rua Meir, 33 |
| Campo | Carla Soares (Campo NOME) | | |

Figura 1.5: Hierarquia de Dados (**Ilustração: trocar a cor cinza por azul claro ou amarela clara, favor não mudar o conteúdo**)

- ✓ Um campo consiste num grupamento de caracteres. Exemplo: o nome. Pode ter também os campos idade ou endereço, por exemplo.

NOME
Carla Soares

- ✓ Um registro consiste num grupo de campos relacionados. Exemplo: o nome, a idade e o endereço da cliente Carla Soares.

Carla Soares 42 Rua Meir, 33

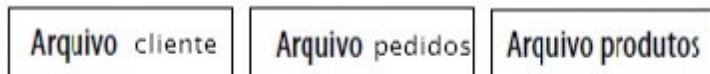
- ✓ Um arquivo consiste num grupo de registros do mesmo tipo. Exemplo: Arquivo cliente.

Arquivo cliente

| NOME | IDADE | ENDEREÇO |
|--------------|-------|---------------|
| Carla Soares | 42 | Rua Meier, 33 |
| Roberto L. | 54 | Rua Beira, 11 |
| João P. | 35 | Rua G, 67 |

- ✓ Um grupo de arquivos é chamado de Banco de Dados. Exemplo: Arquivo cliente, Arquivo pedidos e Arquivo produtos.

Banco de dados vendas



De igual forma, Arquivo de pedidos e Arquivo produtos contêm um grupo de registros e, em cada registro, um grupo de campos.

Agora, vamos mostrar a figura 1.5 como um conjunto de registros dispostos em estruturas regulares, ou seja, como tabelas. Observe a Figura 1.6:

| Arquivo de cliente | | |
|--------------------|-------|---------------|
| Nome | Idade | Endereço |
| Carla Soares | 42 | Rua Meier, 33 |
| Roberto L. | 54 | Rua Beira, 11 |
| João P. | 35 | Rua G, 67 |

| Arquivo de pedidos | | |
|--------------------|---------|------------|
| Nome | Código | Quantidade |
| Carla Soares | 261100 | 12 |
| Roberto L. | 343556 | 40 |
| João P. | 6712399 | 39 |

| Arquivo de produtos | | |
|---------------------|------------------|----------------|
| Código | Nome Produto | Preço Unitário |
| 261100 | Desinfetante | 1,50 |
| 343556 | Cloro | 1,50 |
| 6712399 | Detergente em Pó | 4,60 |

Figura 1.6. Arquivos dispostos como tabelas

(Ilustração: favor trocar as setas, mantendo direções e o conteúdo)

Observa-se que os dados armazenados em linhas são registros e os dados armazenados em colunas são os campos. Os dados de uma tabela normalmente descrevem um assunto específico, pois a figura 1.6. retrata o caso de clientes que fazem compras de produtos através de pedidos.

E o que são essas duas setas ao lado das tabelas?

Você já vai entender: o conceito de tabelas + as setas = Banco de Dados Relacionais. Vejamos o porquê.

Caixa de ênfase

Um Banco de Dados Relacional organiza seus dados em relações. Cada relação pode ser vista como uma tabela pela qual cada coluna corresponde aos campos ou atributos da relação e as linhas correspondem aos registros ou tuplas ou elementos da relação.

Fim da caixa de ênfase

Um conceito importante em um Banco de Dados Relacional é o **atributo chave**, que permite identificar e diferenciar uma tupla de outra. Através do uso de chaves é possível acelerar o acesso a elementos e estabelecer relacionamentos entre as múltiplas tabelas de um Sistema de Banco de Dados Relacional.

Essa visão de dados organizados em tabelas oferece um conceito simples e familiar para a estruturação dos dados, sendo um dos motivos do sucesso dos Sistemas de Banco de Dados Relacionais.

Início do verbete

Atributo chave

Atributo através do qual é possível identificar determinado registro. Uma chave não pode ser repetida, ou seja, o conjunto de valores que constituem a chave deve ser único dentro de uma tabela.

Fim do verbete

Agora que você já conhece os conceitos acima descritos, que tal observar um

exemplo?

Na figura 1. 6. consideremos o atributo chave Nome do Arquivo cliente, pois podemos relacioná-lo com o Arquivo de pedidos. Se o atributo chave fosse Carla Soares no Arquivo de clientes, obteríamos os dados de Idade e Endereço dela. Assim, com o mesmo atributo chave poderíamos relacioná-la com o Arquivo de pedidos, e encontrarmos o atributo Código e o atributo Quantidade da cliente Carla Soares. Por outro lado, não seria possível acessar os dados do Arquivo de produtos pelo atributo chave Nome, pois tal atributo não existe.

Se acessarmos o atributo chave Carla Soares no Arquivo de pedidos, podemos obter os atributos Código e Quantidade pertencentes a ela podendo, através do atributo Código, relacioná-la com o atributo chave Código no Arquivo de produtos e, assim, obtermos os dados do Nome do produto e o Preço Unitário.

Em resumo, podemos relacionar os três arquivos através do atributo chave. No caso da cliente Carla Soares, teremos os seguintes dados:

- Idade: 42 anos;
- Endereço: Rua Meier 33;
- Um pedido de produto com o código 261100;
- Quantidades: 12;
- Nome do produto: desinfetante;
- Preço unitário: R\$1,50.

Agora, imagine o que poderia acontecer se esses Arquivos de Dados estivessem isolados? Suponhamos que o Departamento Pessoal possua o Arquivo cliente feito em Excel, o Departamento de Contabilidade possua o Arquivo pedidos feito em Word e o Departamento Financeiro possua o Arquivo produto feito em Access. Se algum Departamento precisar de alguma informação terá que solicitá-la ao outro Departamento. Mas que confusão, não é mesmo? Além de confuso, muito tempo seria perdido até obter a informação completa sobre o cliente!

Claro que em uma empresa ou instituição real cada Arquivo acima citado possui mais

atributos que o do nosso exemplo. Para evitar que os Arquivos de Dados estejam isolados e feitos em diferentes ambientes, é necessário que façamos uma Integração de Dados.

Veja, a seguir, como a Integração de Dados pode ser útil.

Início Verbete

Integrado

Por integrado, queremos dizer que o Banco de Dados pode ser considerado como uma unificação de vários arquivos.

Fim do verbete

Temos mais uma definição que coloca o **banco de dado** como um conjunto de dados integrados que visa atender um conjunto de aplicações. Por conjunto de aplicações, entendemos as aplicações que são feitas para que qualquer usuário possa utilizar e acessar o **Banco de Dados**.

Exemplo: Um buscador de página Web é uma aplicação que recupera informação de um gigantesco Banco de Dados.

Vamos supor que temos certo tipo de produto e as informações referentes a ele, tais como vendas, compras e produção que estejam armazenadas em arquivos diferentes, como demonstrado na figura 1.7.



Figura 1.7. Dados não integrados

(Ilustração, favor fazer esquema baseado neste)

Nesta forma de armazenamento encontramos problemas de falta de Integração de Dados:

- a) Redundância e Inconsistência dos dados:
 - * Arquivos de formatos diferentes;
 - * Programas desenvolvidos em diferentes linguagens;
 - * Informação repetida em diferentes lugares;
 - * Dados que não representam a realidade.
- b) Dificuldade de extração de informação;
- c) Redundância de rotinas;
- d) Dificuldades no desenvolvimento de novas aplicações.
- e) Problemas com acesso as informações;

Exemplo:

Item a: visto que os arquivos e programas aplicativos são criados por programadores diferentes durante um longo período de tempo, os arquivos provavelmente terão formatos diferentes e os programas serão escritos em diversas linguagens de programação.

Poderemos encontrar o mesmo elemento de informação duplicado em diversos arquivos. Suponha que o atributo código do produto e o atributo nome possam aparecer no Arquivo de Vendas e o no Arquivo de Produção. Essa redundância levará a altos custos de armazenamento e acesso. Se realizarmos a mudança do nome do produto só no Arquivo de Vendas e não no Arquivo de Produção, resultará em uma inconsistência de dados.

Item b: suponha que o gerente de produção necessite encontrar os nomes de todos os produtos produzidos em quantias maiores que 1000 unidades. Caso este tipo de lista não exista ou não tenha sido antecipada quando o sistema original foi projetado, não haverá nenhum programa aplicativo para gerar essa lista. Neste caso, o gerente tem duas saídas: ou ele pega a lista de todos os produtos produzidos e extrai a informação necessária manualmente ou pede ao departamento de processamento de dados que faça tal aplicativo. Ambas as alternativas são obviamente insatisfatórias. Imagine que

tal programa tenha sido de fato escrito e que alguns dias mais tarde, o mesmo diretor necessite destacar da mesma lista apenas os nomes de produtos produzidos com preço de custo maior que R\$ 5000,00. Novamente o gerente tem duas opções e nenhuma é satisfatória. Este tipo de ambiente convencional de processamento de arquivos não permite que os dados necessários sejam recuperados de uma maneira conveniente e eficiente.

Item c: como já mencionado, os programas aplicativos podem ser criados por programadores diferentes em diversas linguagens de programação. Então, diversos programas podem conter rotinas iguais, ou seja, ocasionando duplicidade de rotinas ou processos desnecessários.

Item d: as dificuldades no desenvolvimento de novas aplicações devem-se ao isolamento de dados, visto que os dados estão espalhados em diversos arquivos podendo ter formatos diferentes.

Item e: arquivos isolados não fornecem informação completa do cliente. Se o mesmo cliente fosse cadastrado em uma tabela por várias vezes e necessitássemos atualizá-lo ou excluí-lo, isto teria que ser feito em diversos lugares.

Então, qual a solução para evitar tais problemas? Integração de dados!

Em Bancos de Dados onde há integração dos dados, cada informação (vendas, compras e produção) é armazenada uma única vez, como se observa na figura 1.8.

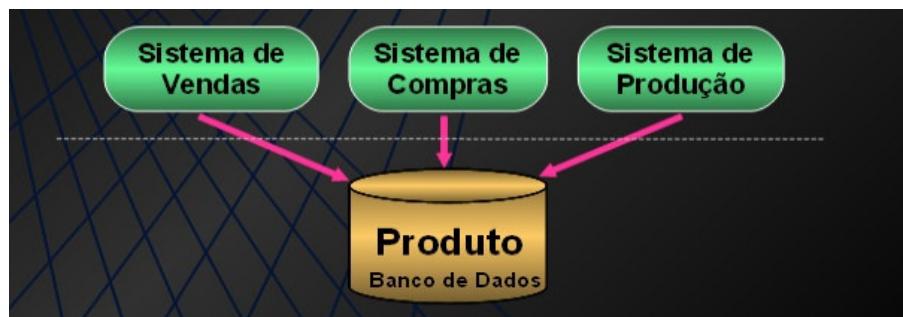


Figura 1.8. Dados integrados

(Ilustração- favor fazer esquema baseado neste)

Por isso, o Banco de Dado é também um conjunto de dados integrados que visa atender um conjunto de aplicações.

O desenvolvimento da teoria de Banco de Dados ajudou, sobretudo, na independência de dados, ou seja, qualquer mudança na estrutura física ou na estratégia de acesso não implica em alteração nos aplicativos que utilizam tal dado.

Início do Box de atenção

Vantagens dos Bancos de Dados Integrados

- Pode reduzir ou eliminar a redundância (repetição de dados);
- A inconsistência (ou seja, dados que não representam a realidade) pode ser evitada (até certo ponto);
- Pode aplicar restrições de segurança: nem todo usuário do Sistema de Banco de Dados deve ter acesso a todos os dados. Por exemplo: num sistema bancário, o departamento de pessoal necessita apenas de parte do Banco de Dados que tenha informações sobre os diversos empregados do banco. Eles não necessitam ter acesso à informação sobre as contas dos clientes do banco;
- Pode manter a integridade dos dados: garante a qualidade dos dados em um Banco de Dados. Por exemplo: se o código de um produto é introduzido com um valor de 0049 em uma tabela chamada produtos, o Banco de Dados não deve permitir que outro produto tenha um código com o mesmo valor.
- Redução da perda de espaço de armazenamento;

Fim do Box de atenção

Você se lembra da figura com os dados sentados num banco? Vamos associar cada dado com um arquivo de dados. Nesse sentido, um grupo de dados representa o conjunto de arquivos. Então temos que os dados juntos sentados no banco – o conjunto de arquivos - são todos os dados integrados num Banco de Dados. Ficou mais fácil de entender?

Um Banco de Dados com dados integrados precisa ser gerenciado e isto é feito através do **Sistema de Gerenciamento de Banco de Dados**.

Representação simplificada de um SBD:

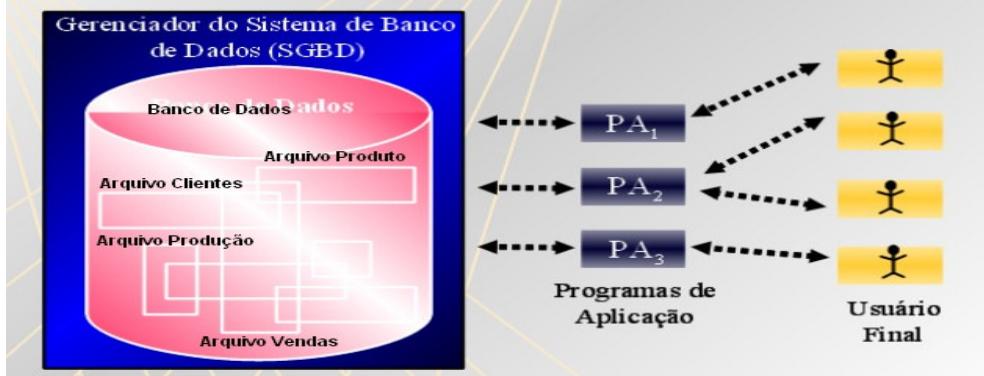


Figura 1.9. O Sistema de Banco de Dados

(Ilustração, favor fazer esquema baseado neste, boneco pode ser diferentes pessoas)

Início de verbete

Sistema de Gerenciamento de Banco de Dados (SGBD)

É o software (coleção de programas) responsável pelo Banco de Dados. Operações como consultas, inserções, atualizações, exclusões, recuperação de dados, armazenamento dos dados, criação de usuários e tabelas, diretivas de segurança, etc. são controladas pelo SGBD. Além disso, o SGBD garante a integridade dos dados, cuida da segurança, retorna consultas, compartilha dados, controla a redundância, as transações, e tudo isso de forma transparente para o usuário.

Fim do verbete

Podemos citar como exemplos de SGBDs: SQL Server, Oracle, Firebird, MySQL, Interbase, entre outros. Estes programas, em geral, são chamados SGBDs relacionais.

A figura 1.9 mostra um Sistema de Banco de Dados que é composto de:

- SGBD,
- Conjunto de Programas de Aplicação,
- Conjunto de Usuário Final.

Mas você sabe quem é o Usuário Final?

Um usuário final pode ser o programador de aplicação (responsável pela escrita de programas de aplicações de Banco de Dados), o usuário que acessa o Banco de Dados a partir de uma aplicação on-line ou o administrador do Banco de Dados.

Início Box Multimídia

Quer conhecer os maiores bancos de dados?

Pesquise e descubra sobre os 10 maiores Bancos de Dados do Brasil em

<http://rankz.wordpress.com/2008/11/04/os-dez-maiores-bancos-do-brasil/>

Pesquise e descubra sobre os maiores Bancos de Dados do mundo em

[http://imasters.uol.com.br/artigo/15432/bancodedados/os maiores bancos de dados do mundo/](http://imasters.uol.com.br/artigo/15432/bancodedados/os-maiores-bancos-de-dados-do-mundo/)

Fim Box Multimídia

Início da atividade

Atividade 2 – Atende ao objetivo 2

Explique, com suas próprias palavras, o que é um Banco de Dados Relacional.

Diagramação, inserir 5 linhas para a resposta.

Resposta Comentada

Define-se como “conjuntos de dados vistos segundo um conjunto de Tabelas, e as operações sobre elas são feitas por linguagens que manipulam o Banco de Dados. Possui um atributo chave que serve para acessar elementos e estabelecer relacionamentos entre as múltiplas tabelas”. Nos termos do Banco de Dados Relacional, uma tabela é conhecida como Relação, campo como atributo e linhas como tuplas. O desafio mais significativo, ao implementar um Banco de Dados, é desenvolver sua estrutura e comportamento corretamente.

Fim da atividade

Início da atividade

Atividade 3 – Atende ao Objetivo 2

Imagine que temos um Banco de Dados com as seguintes informações:

| ARQUIVO DE MATERIAL | | | | | |
|---------------------|-----------------------|----------------|-------------------------|----------------------|-------------|
| CÓDIGO DO MATERIAL | DESCRIÇÃO DO MATERIAL | ESTOQUE MÍNIMO | UNIDADE DE FORNECIMENTO | ESTOQUE DE SEGURANÇA | LOCALIZAÇÃO |
| 001010 | Tipo 1 | 1500 | Local 3 | 180 | RJ |
| 020201 | Tipo 2 | 800 | Local 2 | 250 | SP |

| ARQUIVO HISTÓRICO DE MOVIMENTAÇÃO DO MATERIAL | | | | | |
|---|------------------------|-------------------|------------|-------------|------------|
| CÓDIGO DO MATERIAL | DATA-HORA MOVIMENTAÇÃO | TIPO MOVIMENTAÇÃO | QUANTIDADE | NOTA FISCAL | FORNECEDOR |
| 001010 | 2/6/09 12h | Leve | 320 | 156231 | JOÃO S.A |
| 020201 | 7/9/10 15h | Médio | 480 | 231583 | Lucas Cia |

(Diagramação, favor fazer tabelas como essas, sem mudar o conteúdo)

Responda às seguintes questões baseadas no banco de dados apresentado anteriormente:

- Quantos arquivos compõem o Banco de Dados? E Quais são?
- De quantos campos ou atributos o Arquivo de Material é formado?
- Mencione três atributos do Arquivo Histórico de Movimento do Material.
- Mencione um registro ou tupla de cada Arquivo.
- O Banco de Dados acima é do tipo relacional? Justifique sua resposta.

Diagramação: favor deixar uma linha para cada item.

Resposta Comentada

Além de obter uma definição correta de Banco de Dados, é muito importante ter clareza nos conceitos de arquivos, registros e campos.

Item a: o Banco de Dados está composto por dois arquivos. O Arquivo de Material e o Arquivo Histórico de Movimentação do Material.

Item b: o Arquivo de Material está formado por seis atributos.

Item c: Código do material, Quantidade e Fornecedor.

Item d: o segundo registro do Arquivo de Material :

| | | | | | | | | | | |
|--------|--|--------|--|-----|--|---------|--|-----|--|----|
| 020201 | | Tipo 2 | | 800 | | Local 2 | | 250 | | SP |
|--------|--|--------|--|-----|--|---------|--|-----|--|----|

E o primeiro registro do Arquivo Histórico de Movimentação do Material:

| | | | | | | | | | | |
|--------|--|------------|--|------|--|-----|--|--------|--|----------|
| 001010 | | 2/6/09 12h | | Leve | | 320 | | 156231 | | JOÃO S.A |
|--------|--|------------|--|------|--|-----|--|--------|--|----------|

Item e: Sim. É um Banco de Dados Relacional, pois os dados estão organizados como tabela onde cada coluna corresponde a atributos da relação e cada linha corresponde às tuplas da relação. Possui também um atributo chave chamado Código do material, o qual serve para acessar elementos e estabelecer relacionamentos entre os dois Arquivos.

Fim da atividade

Banco de Dados e Tecnologia de Informação

Banco de Dados é o componente da **Tecnologia de Informação** voltado para o armazenamento da informação a ser utilizada em um processo de tomada de decisão. A estrutura e comportamento de um banco de dados devem propiciar esse armazenamento de forma persistente e consistente.

Início Verbete

Tecnologia de Informação

É o conjunto de recursos não humanos dedicados à comunicação, processamento e armazenamento da informação e a maneira como esses recursos são organizados em um sistema capaz de executar um conjunto de tarefas.

Fim do verbete

Temos então que o Banco de Dados é o componente da Tecnologia de Informação, cuja função é dar suporte ao armazenamento dentro do ciclo de vida da informação. Veja a figura 1.10.

1. Captura da Informação;
2. Transmissão da Informação;
3. Processamento da Informação;

4. Armazenamento da Informação;
5. Exibição da Informação.



Figura 1.10. Ciclo de vida da Informação

(Ilustração: favor desenhar uma imagem baseada nesta, mas trocar de pessoa e computador (pode ser mais de uma pessoa falando), mudar as cores, tirar o balão de pensamento. Apenas manter a cor verde no cilindro, pois ela representa um Banco de Dados)

Em geral, o suporte tecnológico da informação é composto pelas seguintes tecnologias:

1. Tecnologia de Interface Homem-Máquina (Captura e Exibição da Informação);
2. Tecnologia de Transmissão (Redes de Computadores);
3. Tecnologia de Processamento (Centralizado, Distribuído, Paralelo e Cliente-Servidor);
4. Tecnologia de Armazenamento.



Figura 1.11. Suporte Tecnológico da Informação

(Ilustração: desenhar algo semelhante, mas trocar os círculos superiores por dois retângulos e os retângulos trocar por círculos, mudando as cores, mantendo a cor verde em “Tecnologia para Armazenamento”, colocar a seta maior na parte inferior e a seta menor na parte superior.)

Portanto, a Tecnologia de Armazenamento é a Tecnologia de Banco de Dados, encarregada pelo armazenamento (e recuperação) da informação.

A estrutura e o comportamento do Banco de Dados devem propiciar o armazenamento de forma persistente e consistente. Mas isso explicaremos mais adiante.

Box de Curiosidade

Evolução: Tecnologia de Informação em Banco de Dados

- ✗ Até 1960: Sistema de Arquivos integrados. Ex.: ISAM, VSAM;
- ✗ Final de 1960: Modelo Hierárquico. Ex.: IMS(IBM);
- ✗ 1970 e início de 1980: Modelo de Redes. (CODASYL) Ex. : IDMS, DMS-II(Unisys);
- ✗ Meados 1980: Modelo Relacional. Ex.: DB-2, SQL-DS (IBM), Oracle, Ingres;
- ✗ Final de 1980: Modelo Orientado a Objetos e Relacional Estendido (Objeto-Relacional) Ex.: BDOO: Vbase, O2, Orion, Gemstone, Jasmine, ObjectStore. BDOR: Postgres, Informix, Oracle 9i, IBM DB2;
- ✗ 1990: BD Inteligentes e Cliente-Servidor, BD e Web, BD multimídia, BD Espacial, Ativo, Temporal, Dedutivo;
- ✗ 2000: Modelo Objeto Relacional, BD de WWW, BD Mobile, SQL/MM

Fim do Box de Curiosidade

Agora que já vimos diversos conceitos importantes sobre Banco de Dados, vamos colocá-los em prática através das seguintes atividades.

Início da atividade

Atividade 4 – Atende ao Objetivo 2

Já sabemos que quando os dados são armazenados em arquivos diferentes encontramos vários problemas. Quais tipos de problemas são encontrados, usando como base o exemplo de dados a seguir, e como devemos solucioná-los.

Diagramação, inserir 5 linhas para a resposta

Tabela Salário

| Código | Nome | Endereço | Salário | Data-Saida |
|--------|-------|-----------------|---------|------------|
| 1 | João | Rua Bahia 111 | 1200,00 | |
| 2 | Lucas | Rua Coronel 678 | 560,00 | |
| 3 | Mario | Rua Flor 112 | 670,00 | |
| 4 | Ana | Rua F 11189 | 870,00 | 08/04/2010 |

Tabela Status

| Código | Nome | Data-Ingreso | Sexo | Status |
|--------|-------|--------------|------|------------|
| 1 | João | 01/01/2000 | M | Ativo |
| 2 | Lucas | 06/09/1999 | M | Inativo |
| 3 | Mario | 20/12/2005 | M | Ativo |
| 4 | Ana | 30/08/1980 | F | Aposentada |

(Diagramação, favor fazer tabelas como essas, sem mudar o conteúdo nem as cores do lado direito.)

Resposta Comentada

Este tipo de atividade é uma fase prévia da fase de Modelagem de Dados. É importante reconhecer os problemas inerentes por falta de Integração de Dados. Neste caso, observamos que temos duas tabelas.

Note que os nomes dos clientes se repetem em ambas as tabelas, pois cada tabela é importante para cada área ou departamento específico. Observa-se que temos os seguintes problemas:

- ✓ Redundância dos dados - Os funcionários são cadastrados duas vezes. Por cada funcionário novo é digitado seu nome em ambas as tabelas, e isto não pode acontecer. Mas você pode se perguntar: eles estão cadastrados duas vezes porque ambas as tabelas possuem dados diferentes do mesmo funcionário e precisa-se saber de quem são aqueles dados em cada tabela? Sim, mas você pode eliminar o atributo nome de uma tabela e continuará sabendo a quem pertence cada um dos

dados.

- ✓ *Dificuldade de acesso e inconsistência dos dados - Caso haja necessidade de se atualizar os dados de um funcionário, isto terá que ser feito em diversos lugares. Na tabela Status observamos que o funcionário Lucas possui um status de inativo, mas está informação não foi comunicada ao setor que utiliza a tabela Salário, pois nesta tabela não foi preenchido o atributo Data-Saída.*
- ✓ *E também existe um desperdício de espaço físico no disco rígido.*

Solução ==> Integração de dados!

(1) Podemos criar uma tabela incluindo todos os atributos de ambas as tabelas copiando uma só vez o atributo nome.

(2) Podemos eliminar os nomes da primeira tabela e através do código, acessar ambas as tabelas obtendo-o como chave.

Para os dois casos precisamos atualizar o status do funcionário Lucas.

Fim da atividade

Para se ter sucesso... é preciso planejar!

Muitos dos problemas enfrentados no dia-a-dia de qualquer *empresa informatizada* resultam de um projeto “mal feito” ou “mal implementado”. A modelagem de banco de dados é um tema que nunca sai de moda. Independente do sistema, um banco de dados sempre deve ser iniciado por um bom projeto e uma boa modelagem, pois é justamente a partir deste ponto que será garantida a confiabilidade, eficiência e eficácia do sistema.

O Projeto de um Banco de Dados

A modelagem de banco de dados é estudada há mais de três décadas e vem sendo, ao longo desse tempo, definida por diversos autores. Apresentamos as seguintes definições:

- ✓ É o processo que determina a organização de um banco de dados, incluindo a

sua estrutura, conteúdo e aplicações.

- ✓ É o processo de projeto da estrutura lógica e física de um ou mais bancos de dados, visando proporcionar as informações necessárias de um determinado conjunto de aplicações aos usuários de uma organização.

Mas por que devemos usar a modelagem de dados?

- ✓ Proporciona informação concisa dos dados necessários para o negócio;
- ✓ Modelo de Dados independe de hardware e software;
- ✓ Identifica redundâncias;
- ✓ Identifica dados não utilizados;
- ✓ Permite estabelecer e manter regras de integridade dos dados;
- ✓ É um auxiliar precioso na concepção do banco de dados;
- ✓ É fundamental para um bom desempenho do sistema;
- ✓ Os dados devem estar organizados de modo a serem flexíveis para responder às necessidades dos usuários.

Obter dados com estas características é o principal objetivo da modelagem de dados. A modelagem do banco de dados é extremamente importante para que o sistema tenha uma boa performance.

Estudos indicam que quanto maior o tempo despendido no Projeto de Banco de Dados, menor será o tempo utilizado em sua manutenção. Se a devida atenção não for dada ao desenho do banco de dados, todo o desenvolvimento do sistema poderá ser comprometido. Um banco de dados bem projetado fornece um acesso conveniente a todas informações desejadas. Com uma boa estrutura, assegura-se resultados mais rápidos e precisos.

O Projeto de um Banco de Dados é um processo complexo que visa atingir algumas metas:

- ✗ Satisfazer os requisitos de informações especificadas por usuários e aplicações;
- ✗ Proporcionar uma estruturação natural e fácil para entender a informação;
- ✗ Dar suporte a quaisquer requisitos de processo e objetivos de desempenho,

como tempo de resposta, tempo de processamento e espaço de armazenamento.

Para o cumprimento das metas mencionadas, o processo é dividido em diferentes fases: Projeto Conceitual, Projeto Lógico e Projeto Físico. Esta complexidade é melhor gerenciada quando o problema é subdividido em vários subproblemas independentes. O fato central nas três fases do Projeto de Banco de Dados é a modelagem do dado e suas propriedades.

Projeto (Modelagem) de Banco de Dados

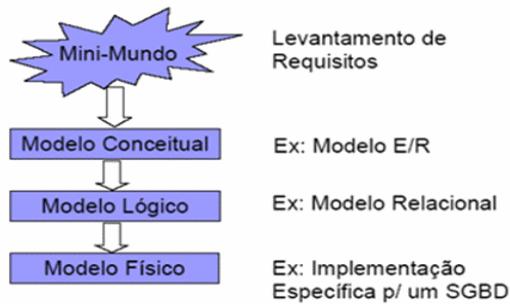


Figura 2.7. Projeto ou Modelagem de Banco de Dados

Modelo conceitual: É também uma descrição da estrutura do Banco de Dados, porém, de ~~uma~~ forma independente da implementação que será feita. Em outras palavras, ele independe de qual SGBD será utilizado na implementação. O Modelo Conceitual indica quais os dados que poderão aparecer no Banco de Dados, mas não informa de que forma estes mesmos dados serão armazenados no nível SGBD.

Modelo lógico: Este modelo descreve a estrutura do Banco de Dados no nível do usuário SGBD, neste caso, o administrador de dados. Este modelo já se aproxima mais da implementação que será feita, ou seja, torna-se dependente de qual SGBD será implementado.

Modelo físico: Este modelo é o último passo antes da geração de *scripts* de

implementação. Ele é totalmente dependente do SGBD específico que será utilizado. Além das definições de chave (que já estão presentes no modelo lógico), o modelo físico contempla definições de armazenamento que não têm influência alguma nas etapas anteriores, mas que são essenciais no tocante à performance geral do Banco de Dados.

Conclusão

Os Bancos de Dados Relacionais são, atualmente, o tipo Banco de Dados mais utilizado no mundo. Um Banco de Dados Relacional organiza seus dados em *relações*. Cada relação pode ser vista como uma tabela pela qual cada coluna corresponde aos campos ou *atributos* da relação e as linhas correspondem aos registros, às *tuplas* ou elementos da relação.

Um conceito importante em um Banco de Dados Relacional é o conceito de atributo chave, que permite identificar e diferenciar uma tupla de outra. Através do uso de chaves é possível acelerar o acesso a elementos e estabelecer relacionamentos entre as múltiplas tabelas de um Sistema de Banco de Dados Relacional. O desafio mais significativo ao implementar um Banco de Dados é desenvolver sua estrutura e comportamento corretamente.

Resumo

Vamos rever os principais conceitos vistos nesta aula:

- Dado: é qualquer elemento identificado em sua forma bruta que, por si só, não conduz a uma compreensão de determinado fato ou situação;
- Informação: é o componente que, efetivamente, transita entre os processos organizacionais e, dessa forma, estabelece relacionamentos entre os mesmos;
- Podemos definir o termo Banco de Dados como: "Um conjunto de arquivos relacionados". O termo arquivo é relacionado como uma tabela pela qual as colunas são campos e as linhas são os registros;
- Um Banco de Dados Relacional é aquele que: 1. Organiza seus dados como

uma tabela pela qual cada coluna corresponde a atributos da relação e cada linha corresponde às tuplas ou elementos da relação e 2. Possui um atributo chave que serve para acessar elementos e estabelecer relacionamentos entre as múltiplas tabelas;

- O papel da Tecnologia da Informação é dar suporte à Informação em todo o seu ciclo de vida compreendendo: Captura, Transmissão, Processamento, Armazenamento e Exibição da Informação:

Dentro desse contexto, Banco de Dados é o componente da Tecnologia da Informação voltado para o Armazenamento da Informação;

- O projeto de banco de dados é constituído de 3 fases: modelagem conceitual, lógica e física.

Informações sobre a próxima aula

Na próxima aula, iremos estudar mais a fundo o modelo conceitual, verificando como identificar os elementos mais importantes para iniciar um projeto de banco de dados.

Referências Bibliográficas

Date C.J. , 20034. Introdução a Sistemas de Bancos de Dados. 8ed. Americana. Rio de Janeiro. Elsevier.

Setzer V.W. & Corrêa da Silva F.S. 2005. Bancos de Dados. 1ed. São Paulo. Edgard Blucher.

Gurovitz H. O que cerveja tem a ver com fraldas? Site:

<http://www.datawarehouse.inf.br/Artigos/cervejaefraldas.pdf>, acessado 28/09/2010.

Curso de Extensão: Modelando e Implementando Bancos de Dados Relacionais
Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 2

Projeto conceitual – ER e DER

Meta

Apresentar conceitos importantes sobre projeto conceitual de banco de dados e as ferramentas do Modelo ER e DER.

Objetivos

Esperamos que, ao final desta aula, você seja capaz de:

1. Reconhecer como um banco de dados é projetado;
2. Aplicar os conceitos básicos do Modelo Entidade Relacionamento (ER), tais como entidades e seus atributos (incluindo o identificador), relacionamento entre as entidades e seus respectivos atributos;
3. Elaborar o Diagrama de Entidade e Relacionamento (DER).

Pré-requisitos

Para o completo aprendizado desta aula, é importante você relembrar os conceitos de banco de dados relacional, de chave e de projeto conceitual, vistos na Aula 1.

Para ter sucesso... É preciso planejar!

Muitos dos problemas que enfrentamos no dia a dia resultam de um projeto “malfeito” ou “mal-implementado”, não é verdade? Diversas decisões que precisamos tomar devem ser planejadas, como trocar de residência, escolher uma escola para matricular o filho, trocar o carro...

Quando falamos de banco de dados isso não é diferente. Independente do sistema, um banco de dados deve sempre ser iniciado por um bom projeto, pois é justamente a

partir desse ponto que será garantida a confiabilidade, eficiência e eficácia do sistema.

Se a devida atenção não for dada ao desenho do banco de dados, todo o desenvolvimento do sistema poderá ser comprometido. Um banco de dados bem projetado fornece um acesso conveniente a todas as informações desejadas. Uma boa estrutura assegura resultados mais rápidos e precisos.

Esta aula tem como propósito introduzir os principais conceitos de projeto de banco de dados e do Modelo Entidade Relacionamento. A ideia é mesclar os conceitos básicos apresentados nesta aula com a realidade, para ilustrar melhor cada situação em que são desenvolvidos nas atividades práticas. Siga em frente e passe todos os desafios em cada atividade.

Fim da introdução

Projetando um banco de dados

Um banco de dados bem projetado fornece acesso conveniente às informações desejadas de forma rápida e precisa. Mas você sabe como é projetado um banco de dados?

Um banco de dados é projetado, construído e manipulado com dados para um propósito específico, pois possui um conjunto predefinido de usuários e aplicações. Ele representa aspectos importantes ou relevantes do mundo real. Para que possamos construir um bom banco de dados, precisamos verificar quais são esses aspectos. Isso pode ser feito a partir da análise de um texto que consiste em uma “versão resumida” do mundo real ao qual chamamos **“minimundo”**.

Qualquer alteração efetuada no minimundo é automaticamente refletida no banco de dados. Pelo minimundo assimilam-se os requisitos de informação e as regras de negócio, obtendo, assim, domínio da solução. Esse domínio da solução está ligado à estrutura e comportamento do banco de dados, que, por sua vez, está ligado ao **conhecimento organizacional ou de negócio**.

Início do verbete

Minimundo

Descrição representativa de uma parcela do mundo real a partir da qual se pretende absorver o conhecimento organizacional, analisar os problemas existentes, propor alternativas, projetar e implementar uma solução e, posteriormente, implantar, operar e manter em funcionamento essa solução. Minimundo equivale a domínio do problema, domínio de conhecimento ou universo/domínio de discurso.

Conhecimento organizacional ou conhecimento de negócio

É formado pelos requisitos de informação e as regras de negócio. Essas regras refletem nossa necessidade de explicar, restringir, prever (planejar) o comportamento ou características comportamentais dos objetos (coisas, entidades, elementos) do mundo real.

Fim do verbete

Na Figura 2.1 você pode observar a evolução do modelo inicial (Modelo 1) baseado num minimundo até atingir o modelo final (Modelo n) ou domínio da solução. Da passagem do modelo inicial até o modelo final, aplicaram-se técnicas ou ferramentas na busca pelo melhor modelo que representasse o minimundo.

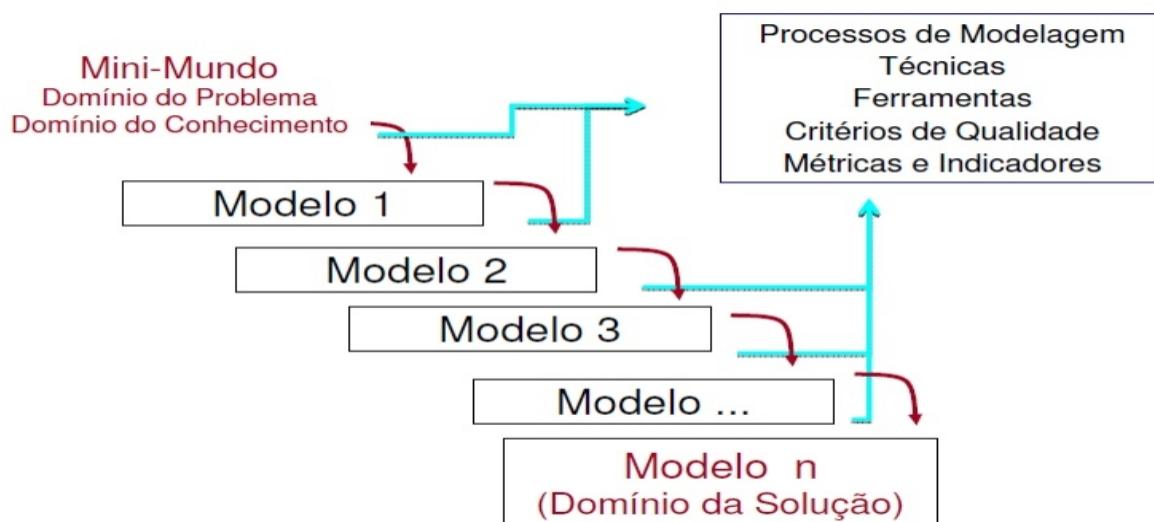


Figura 2.1: Minimundo e domínio da solução.

Você pode observar, na Figura 2.2, que todo e qualquer objeto pertencente à realidade possui estrutura e comportamento. Para iniciar a interpretação de um minimundo, precisamos analisar a sua descrição, de modo a identificar dentro dela aspectos ou objetos relevantes e, então, extrair sua estrutura e comportamento. As características estruturais dos objetos estão relacionadas aos requisitos de informação; o comportamento do objeto está associado às regras de negócio. Os requisitos de informação e as regras de negócio são desenvolvidos dentro de um domínio do problema.



Figura 2.2. Requisitos de informação e regras de negócio.

Você já viu que todo e qualquer objeto pertencente à realidade possui estrutura. No exemplo da Figura 2.3, o objeto a ser considerado é uma pessoa. Então, as informações decorrentes das características estruturais podem estar formadas por: altura, peso, cor, nome, sexo, temperatura e pressão arterial. Portanto, temos que os requisitos de informação expressam as informações decorrentes das características estruturais (reais/abstratas) dos objetos de um domínio de problema necessárias ao controle desses objetos.

REQUISITOS DE INFORMAÇÃO ELEMENTO [OBJETO-ENTIDADE]

Todo e qualquer elemento da Realidade possui uma estrutura e um comportamento.

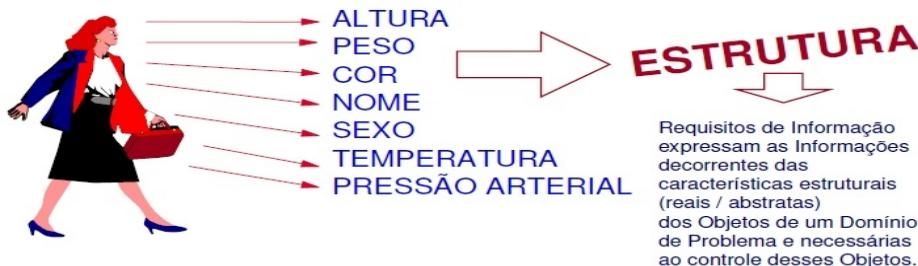


Figura 2.3. Requisitos de informação.

Além de possuir estrutura, todo e qualquer objeto pertencente à realidade também possui comportamento. Ainda no exemplo do objeto como uma pessoa, na Figura 2.4, observa-se que o comportamento pode ser expresso como nascer, crescer, desenvolver, procriar e morrer. Portanto, as regras de negócio expressam a necessidade de explicar, restringir e planejar o comportamento dos objetos de um domínio de problema.

REGAS DE NEGÓCIO ELEMENTO [OBJETO-ENTIDADE]

Todo e qualquer elemento da Realidade possui uma estrutura e um comportamento.



Figura 2.4. Regras de Negócio

Na figura a seguir temos todo o contexto de como um banco de dados é projetado. No exemplo, você pode perceber que o minimundo está associado aos empregados e aos departamentos de certa empresa.

No banco de dados existem duas tabelas (Empregado e Departamento) que representam o modelo descritivo no minimundo. Qualquer alteração efetuada no minimundo é automaticamente refletida no banco de dados. Suponha, então, que o empregado B, que trabalha no departamento D10, mude para o departamento D30. Essa mudança vai se refletir também no banco de dados, resultando em uma mudança no atributo chave Cod-Dep-Trab da tabela Empregado e no atributo chave Cod-Dep na tabela de Departamento.

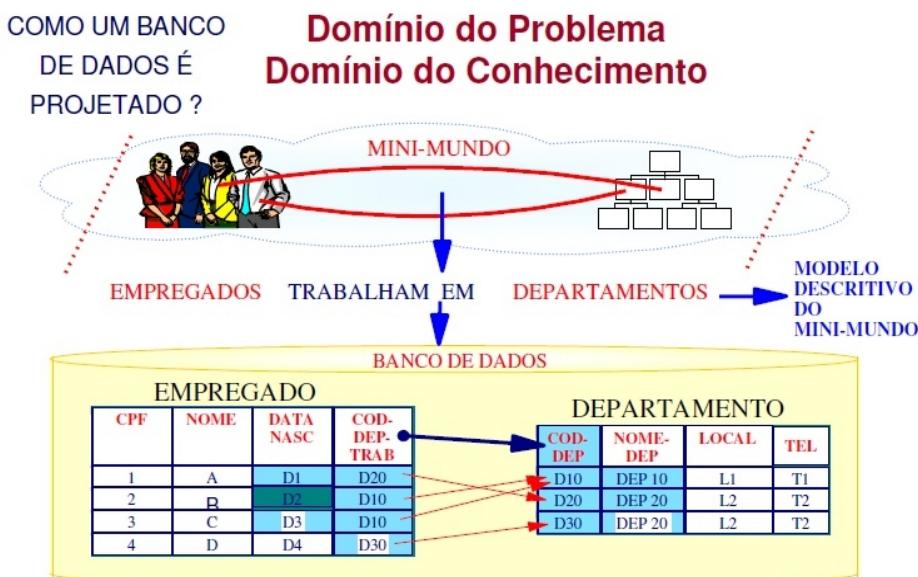


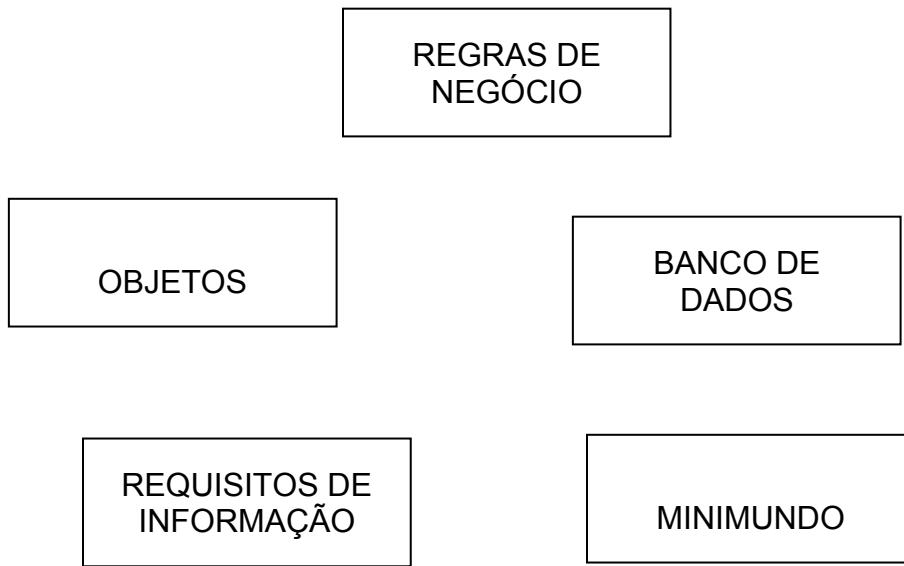
Figura 2.5. Como um Banco de Dados é projetado?

Agora que já revisamos diversos conceitos sobre como um banco de dados é projetado, coloque-os em prática na atividade seguinte.

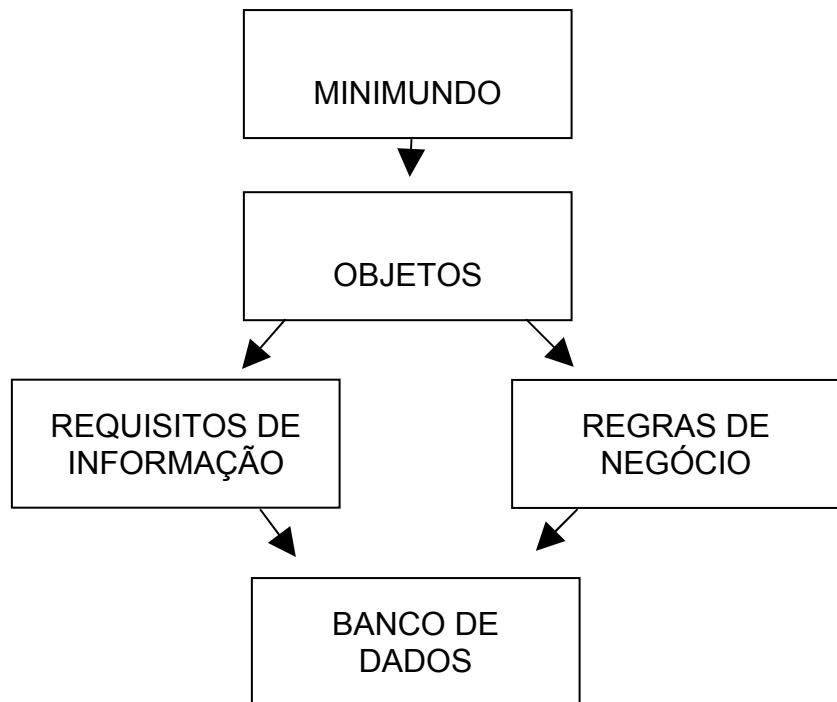
Início da atividade

Atividade 1 - Atende ao objetivo 1

Usando as palavras MINIMUNDO, OBJETOS, REQUISITOS DE INFORMAÇÃO, REGRAS DE NEGÓCIO e BANCO DE DADOS, faça relações entre esses conceitos ligando as caixas a seguir com setas, de forma a ilustrar como um banco de dados é projetado. Em seguida, a partir da figura que você criou, descreva com suas palavras como um banco de dados pode ser projetado.



Resposta comentada



Um banco de dados é projetado conhecendo o minimundo pelo qual é representado, isto é, algum aspecto do mundo real em particular. Os objetos interagem com o minimundo pelos requisitos de informação e pelas regras de negócios. Por isso, é importante distinguir tais conceitos para lidar com qualquer situação e poder projetar um banco de dados.

Fim da atividade

Criando um banco de dados relacional!

Para criar um banco de dados, precisamos em primeiro lugar analisar o minimundo e extrair dele os aspectos ou objetos relevantes. Dessa forma, poderemos criar o modelo conceitual a partir dos dados referentes ao negócio (problema em questão para o qual se deseja realizar uma modelagem).

Razões para a criação do modelo conceitual:

- ✗ Descreve exatamente as informações necessárias ao negócio. Para a modelagem, todas as regras de negócio deverão ser conhecidas, assim como a realização do levantamento de requisitos;
- ✗ Ajuda a prevenir erros do futuro sistema;
- ✗ É a base para o projeto lógico e físico.

O primeiro passo para a construção do modelo conceitual é a realização do levantamento de requisitos, juntamente com a obtenção da regra de negócio. Esta etapa é muito importante, pois a construção do modelo Entidade Relacionamento (ER) depende desse modelo descritivo, que será desenvolvido na segunda etapa desta aula.

Início do Box de curiosidade

Modelo ER (Entidade e Relacionamento) e DER (Diagrama ER)

Foram descritos pela primeira vez por Chen, em 1976. Servem, ainda hoje, como ferramenta para a construção do projeto conceitual de dados. Os modelos percebem o mundo como sendo um conjunto de entidades, atributos e relações entre entidades.

Fim do Box de curiosidade

Os objetivos de uma modelagem ER são:

- ✓ Obter todas as informações requeridas sobre o negócio antes de sua implementação, tornando claras suas regras de funcionamento (regras de negócio);
- ✓ Facilitar a criação do projeto do banco de dados, possibilitando a especificação de sua estrutura lógica.

O modelo ER pode ser visto como um conjunto de símbolos gráficos que representam entidades ou objetos e atributos. No DER (Diagrama ER) é incluído o relacionamento, que é representado por um losango ligado por linhas às entidades (retângulos). Vamos, nesta viagem, à criação de um banco de dados relacional.

Projeto conceitual de banco de dados

Para planejar um banco de dados, devemos começar pelo princípio. Como assim? Construindo o modelo conceitual. Conforme você viu na Aula 1, o modelo conceitual é um modelo de dados abstrato que descreve a estrutura de um banco de dados, independente de um SGBD.

Então agora vamos construir o modelo conceitual. Siga as seguintes etapas:

Etapa 1: Levantamento dos requisitos

Para começar, devemos realizar o levantamento dos requisitos, também chamado descrição de requisitos. Esta etapa consiste em descrever textualmente as necessidades ou expectativas dos usuários do banco de dados. Trata-se de uma pequena história usando uma linguagem informal sobre todos os aspectos relacionados às necessidades de dados e do negócio que se pretende modelar. Um levantamento de requisitos limitado poderá produzir, a médio ou longo prazo, prejuízos à continuidade na criação do banco de dados. A descrição textual produzida como resultado do levantamento dos requisitos é o que chamamos minimundo.

Além da história, poderão ser adicionadas figuras, diagramas com a representação do

sistema ou quaisquer outros recursos que possam ilustrar como é feito o funcionamento do sistema. Assim, a história e os diagramas servirão de auxílio e de entendimento global para imaginar como deverá ser o banco de dados, ou seja, o formato dos dados que pretendemos armazenar.

Etapa 2: Iniciando a modelagem

Usaremos o modelo de análise chamado modelo ER (E de entidade e R de relacionamento).

O modelo Entidade Relacionamento, também chamado Entidade Associação, é usado na maioria dos métodos e ferramentas de auxílio à concepção de banco de dados. A ideia fundamental desse modelo é conservar os conceitos genéricos (objetos, associação, propriedade) usados no processo de abstração, que vai da observação de uma realidade à sua descrição.

Esse modelo analisa o mundo real utilizando os conceitos de entidade e relacionamento. Vamos explicar!

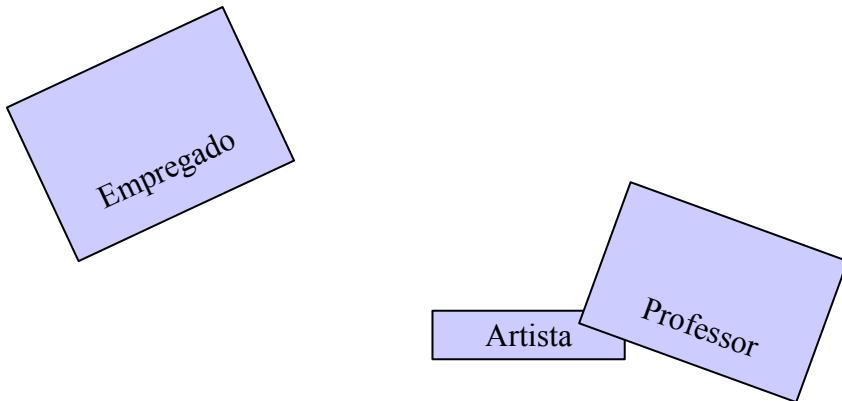
Entidade: representa um objeto de importância para o negócio existente na realidade a ser modelada. O objeto pode ser concreto ou abstrato.

Exemplo:

Entidade ==> objeto concreto: DVD, ator, artista, cantor, professor, aluno, escola etc.

==> objeto abstrato: categoria de CD, gênero do artista etc.

Esse tipo de entidade é representada por um retângulo:



Classe de entidades: representa entidades similares (com as mesmas características).

Exemplo: Trabalhadores, artigos, contratos, departamentos, estudantes etc.

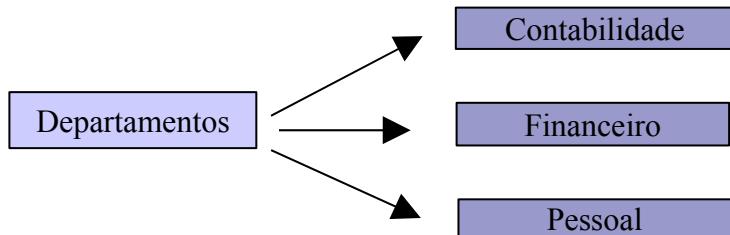


Ilustração: é preciso melhorar os esquemas.

Explicando de outra forma: uma entidade pode ser um computador e a classe de entidade é um conjunto de computadores.

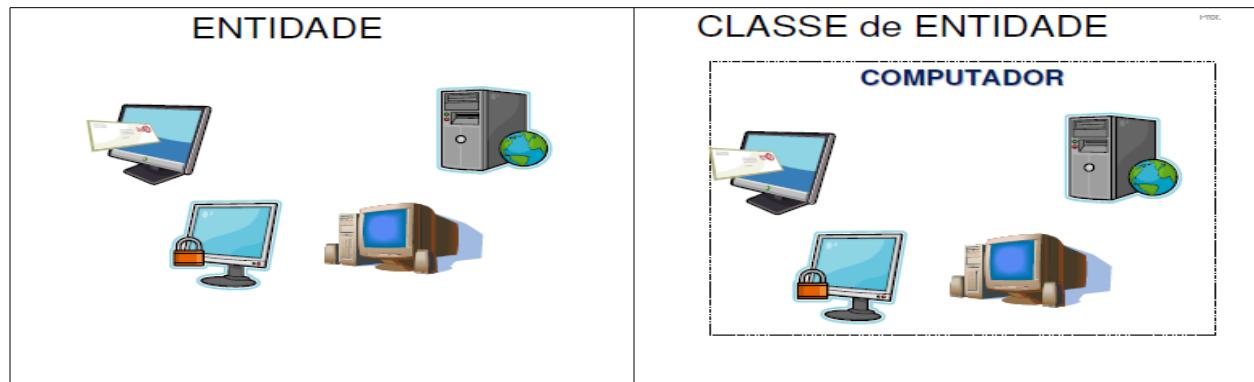


Figura 2.6: Entidade e classe de entidade

Início da caixa de ênfase

- ✖ Uma entidade pode ser representada por um evento (concreto ou abstrato) que se torna de interesse dentro de uma determinada realidade;
- ✖ A toda entidade podem ser associados dados, relacionamentos, atributos etc.
- ✖ Uma entidade é descrita por um conjunto de atributos ou propriedades particulares que a descrevem.

Fim da caixa de ênfase

Como fazer a identificação das entidades pelo modelo descrito obtido por meio do levantamento de requisitos?

São várias as maneiras de iniciar a modelagem de uma realidade, mas uma das mais

interessantes (e adotada por muitos) é analisar a descrição dos requisitos e "pinçar" as informações relevantes. Essa técnica facilita muito a descoberta das principais entidades a serem modeladas; essas são as primeiras entidades desenhadas no modelo ER. Logo, você terá que refinar um pouco mais o levantamento de requisitos para encontrar outras possíveis entidades para o modelo ER.

Atributos ou valores: são propriedades de uma entidade. Podem quantificar, qualificar e classificar uma entidade. Normalmente uma entidade possui vários atributos. Interessa, em termos de modelagem conceitual, que esses atributos representem informações relevantes ao negócio.

Exemplo:

| Entidade | Atributo |
|------------------|--|
| CD | Código, título, nome do cantor, quantidades de CDs. |
| Empregado | Código de matrícula, nome, endereço, departamento, salário, data de nascimento. |
| Carro | Número de placa, tipo, modelo, ano, cor, preço. |
| Tarefa | Código, departamento, responsável, valor/hora, descrição. |
| Pagamentos | Valor, método de pagamento, comentário. |
| Reservas | Data de reserva, hora de reserva, data de entrada, data de saída, data de pagamento, total do pagamento, comentários. |
| Clientes (hotel) | Código, primeiro nome, sobrenome, data de nascimento, tipo, número, complemento, CEP, cidade, estado, país, telefone de contato. |

Início da caixa de ênfase

- ✗ Ao considerarmos o atributo idade da entidade Empregado, não estaremos fazendo uma boa escolha. O ideal seria data de nascimento, ficando o cálculo da idade para quando for necessário. O armazenamento da informação idade é de difícil, ou melhor, impossível atualização;
- ✗ O atributo tamanhos de uniforme do empregado dependerá das regras de negócio. Se a finalidade da entidade Empregado for armazenar dados sobre funcionários numa empresa que forneça uniforme de trabalho, o atributo é coerente. Se a empresa não possui essa política, ele é desnecessário;

- Uma importante decisão precisa ser tomada em relação ao armazenamento de uma informação como um atributo ou uma entidade. Suponha o caso do atributo "Cidade" de clientes de um hotel: ele terá a cidade na qual o cliente reside ou seu país de origem. Se cidade fosse uma entidade, alguns possíveis atributos seriam: nome do país, população e área. Aqui novamente a escolha passa pelas regras de negócio. Em geral, uma informação será um atributo se for de natureza atômica, ou seja, uma característica que faz parte de uma entidade e que, junto com outras informações, ajude a definir essa entidade. Será uma entidade quando possuir uma importância maior, ou seja, for composta de informações que possam (ou necessitem) ser relacionadas a outras entidades.

Fim da caixa de ênfase

Um atributo é representado da seguinte forma:

Atributo normal:

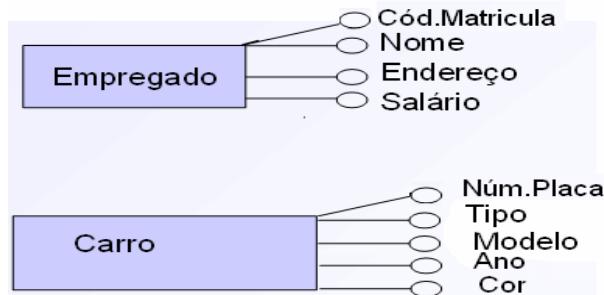


Figura 2.7. Entidade e atributo normal.

Atributo composto: possui mais de uma componente num só atributo.

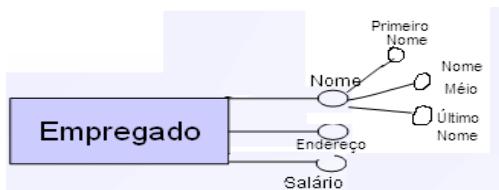


Figura 2.8. Entidade e atributo composto.

O próximo passo é definir os relacionamentos de cada entidade. Para isso, voltamos novamente à descrição dos requisitos para uma nova análise, agora "pincelando" as

relações existentes entre as entidades.

Relacionamento: é uma associação entre entidades. Exemplo: sejam as entidades pessoa e computador:

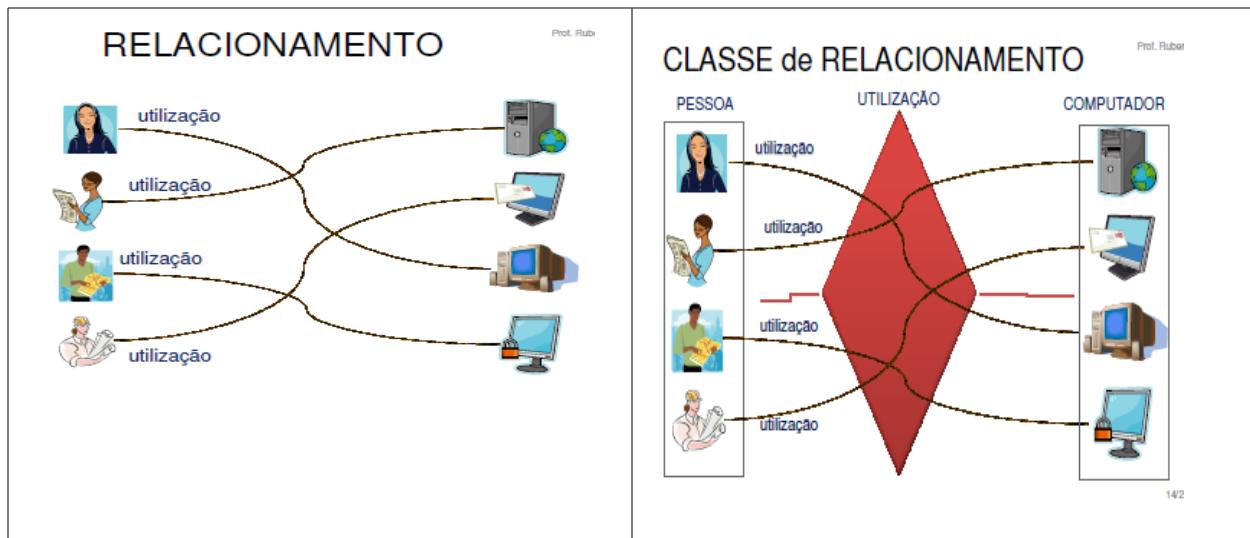


Figura 2.9. Relacionamento e classes de relacionamento.

Os relacionamentos são representados por um losango e linhas que ligam as entidades.

Uma associação (linhas) liga várias entidades; cada uma delas ocupa um "papel".

Se a associação liga duas (ou mais) entidades de um mesmo tipo, ela é dita cíclica, e neste caso a especificação do papel de cada entidade torna-se indispensável.

Entidade A - está associada (através do nomeDoRelacionamento) à entidade B;

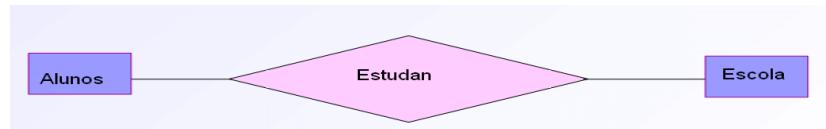
Entidade B - está associada (através do nomeDoRelacionamento) à entidade A.

Graficamente, isso fica representado da seguinte forma:

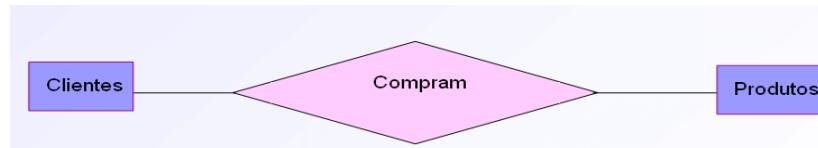


Exemplos: Dados os seguintes enunciados, vamos construir seus diagramas de Entidades:

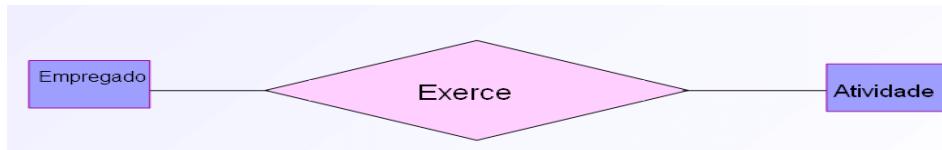
1) Alunos estudam na escola



2) Clientes compram produtos



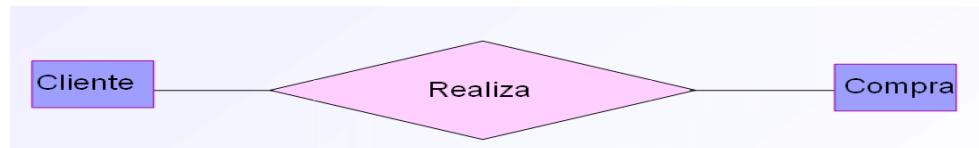
3) Dados os seguintes diagramas de entidade, faça a respectiva interpretação:



Isto quer dizer:

| Entidade | Relacionamento | Entidade |
|------------------------|--------------------------|------------------------|
| Empregado Atividade | exerce é exercida por | Atividade Empregado |

4)



Isto quer dizer:

| Entidade | Relacionamento | Entidade |
|-------------------|----------------------------|-------------------|
| Cliente Compra | realiza é realizada por | Compra Cliente |

Os exemplos anteriores serviram para que fossem apresentadas algumas questões referentes aos relacionamentos entre duas entidades:

1. Todo empregado DEVE (ou PODE) exercer uma atividade?
2. Toda atividade DEVE (ou PODE) ser exercida por um empregado?

Nas duas questões a análise do relacionamento é de EXISTÊNCIA (DEVE – obrigatório ou PODE – opcional).

Logo, temos vários tipos de relacionamento que serão analisados mais à frente nesta aula.

Identificador: também conhecido como chave. É o conjunto de um ou mais atributos ou relacionamentos cujos valores servem para distinguir uma ocorrência da entidade das demais ocorrências da mesma entidade.

Exemplos:

(1) A entidade Cliente possui os atributos CPF, carteira de identidade, nome, endereço, data de nascimento e telefone para contato. Os atributos CPF ou carteira de identidade representam, UNICAMENTE, um cidadão brasileiro, podendo ter a seguinte representação:

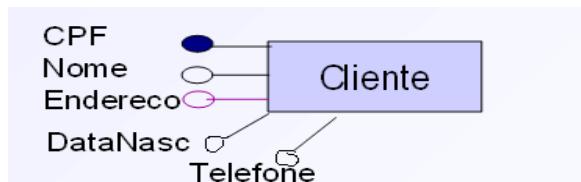


Figura 2.10: Atributos da entidade Cliente.

O atributo identificador ou atributo chave deste exemplo é o atributo CPF.

(2) No caso da entidade Empregado os atributos são: código de matrícula, nome, endereço e salário. O atributo código de matrícula identifica, UNICAMENTE, um empregado. Isso porque no atributo nome pode acontecer de duas pessoas terem nomes iguais; no atributo endereço, se a empresa permitir, pode ter familiares trabalhando na mesma empresa e o endereço vai ser repetido; e o atributo salário pode ser repetido, se estivermos tratando de salário por categorias. Sendo assim, o atributo

identificador é o atributo código de matrícula. De igual forma, podemos escolher, na entidade carro, o atributo identificador número de placa, pois aquele atributo identifica unicamente um determinado carro.

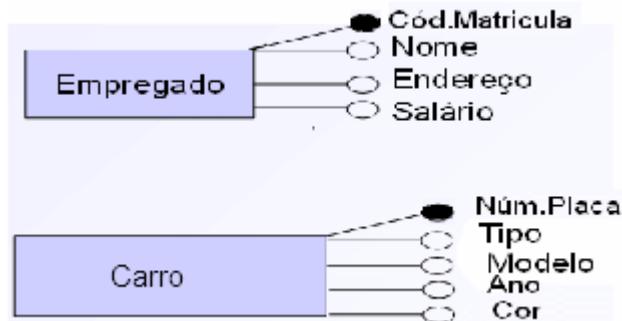


Figura 2.11: Atributos das entidades Empregado e Carro.

Início da atividade

Atividade 2 - Atende ao objetivo 2

Identifique possíveis entidades nos seguintes sistemas:

- a) Sistema transporte
- b) Sistema bancário
- c) Sistema de controle de produção de uma indústria
- d) Sistema cardiovascular
- e) Sistema Via Láctea;

Resposta comentada

Deixando livre a nossa imaginação, podemos considerar as seguintes entidades:

- a) Avião, barco, carro, helicóptero, trem etc.
- b) Cliente, conta corrente, conta poupança, agência, empréstimo etc.
- c) Produto, departamento, estoque, empregado etc.
- d) Coração, artérias, veias, células etc.
- e) Estrela, Sol, buraco-negro, radiação eletromagnética, poeira interestelar etc.

Fim da atividade

Início da atividade

Atividade 3 - Atende aos objetivos 2 e 3

Analise o minimundo apresentado a seguir e liste as entidades envolvidas e seus respectivos atributos. Em seguida, construa um diagrama ER apresentando os relacionamentos existentes entre as entidades que você encontrou.

Minimundo 01

O colégio “Santo Qi” é uma escola particular que oferece formação para o ensino fundamental e médio. Atualmente, está expandindo sua área de atuação, criando uma nova filial. Para tal, o diretor deseja que seja criado um novo sistema que permita substituir as fichas de papel e planilhas Excel empregadas atualmente, de forma a melhorar a organização e agilizar o atendimento administrativo no colégio.

Em conversa com o responsável pela criação do sistema, ele informou que será importante armazenar as seguintes informações sobre os alunos: matrícula, CPF, RG, nome, endereço e data de nascimento. Também lembrou que será necessário informar em qual turma o aluno está inscrito e em quais disciplinas o aluno está matriculado.

Para o cadastro das turmas, será necessário armazenar os seguintes dados: código da turma, quantidade de alunos, horário (manhã ou tarde), ano e ensino (fundamental ou médio).

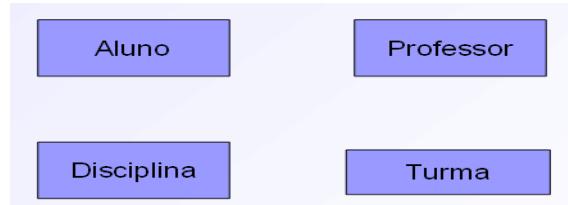
O sistema também deverá ser capaz de emitir as seguintes listagens:

1. Relação do corpo docente (professores) do colégio, contendo: nome do professor, matrícula do professor, data de nascimento, CPF, data de admissão, identificação funcional, PIS/PASEP e tipo de vínculo empregatício;
2. Listagem das disciplinas oferecidas pelo colégio, apresentando, para cada disciplina: código da disciplina, nome da disciplina, descrição da matéria, categoria e créditos;
3. Relatório das disciplinas ministradas por cada professor, exibindo: nome do professor e nome da disciplina que o professor leciona no colégio;
4. A partir da digitação no sistema de uma determinada matrícula de aluno, o sistema deverá ser capaz de emitir uma lista contendo o nome de cada uma das disciplinas que esse aluno está cursando.

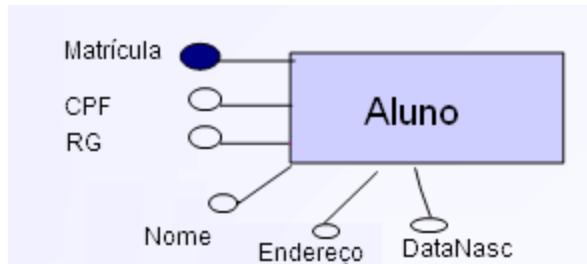
Resposta comentada

Após analisarmos o *minimundo*, efetuamos os seguintes passos:

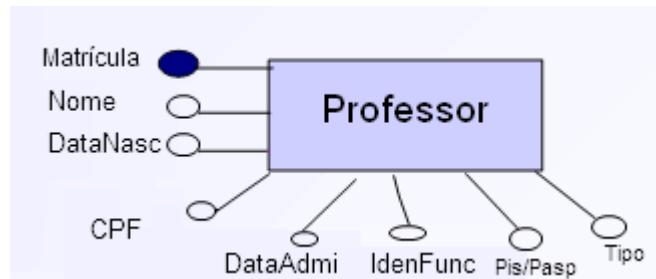
Passo 1: Identificar as entidades com seus respectivos atributos, incluindo o atributo identificador



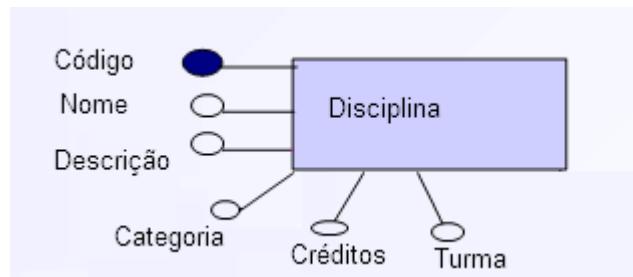
Entidade Aluno: Pode ser identificada a partir do segundo parágrafo do texto do *minimundo*. Possui os seguintes atributos: matrícula (número sequencial conforme a inscrição na instituição), CPF, RG, nome, endereço, data de nascimento. Quaisquer dos atributos matrícula, CPF ou RG podem ser considerados atributo identificador, pois eles identificam um ÚNICO aluno.



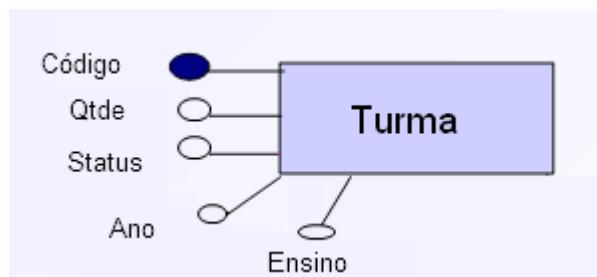
Entidade Professor: Pode ser reconhecida a partir do item 1 solicitado no *minimundo*. Possui os seguintes atributos: número de matrícula, nome, data de nascimento, CPF, data de admissão, identificação funcional, PIS/PASEP, tipo de vínculo. Os atributos número de matrícula, CPF, identificação funcional ou PIS/PASEP podem ser considerados atributo identificador, pois identificam um ÚNICO professor.



Entidade Disciplina: Pode ser reconhecida a partir do item 2 solicitado no minimundo. Possui os seguintes atributos: código da disciplina, nome da disciplina, descrição matéria, categoria, créditos, turma. O atributo identificador é o código da disciplina, pois cada código identifica UNICAMENTE uma disciplina.



Entidade Turma: Pode ser identificada a partir do segundo parágrafo do texto do minimundo. Possui os seguintes atributos: código da turma, quantidade, status (manhã, tarde), ano, ensino (fundamental ou médio). Podemos escolher o Código da turma como atributo identificador, pois ele identifica uma ÚNICA turma.



Passo 2: Identificar os relacionamentos entre as entidades

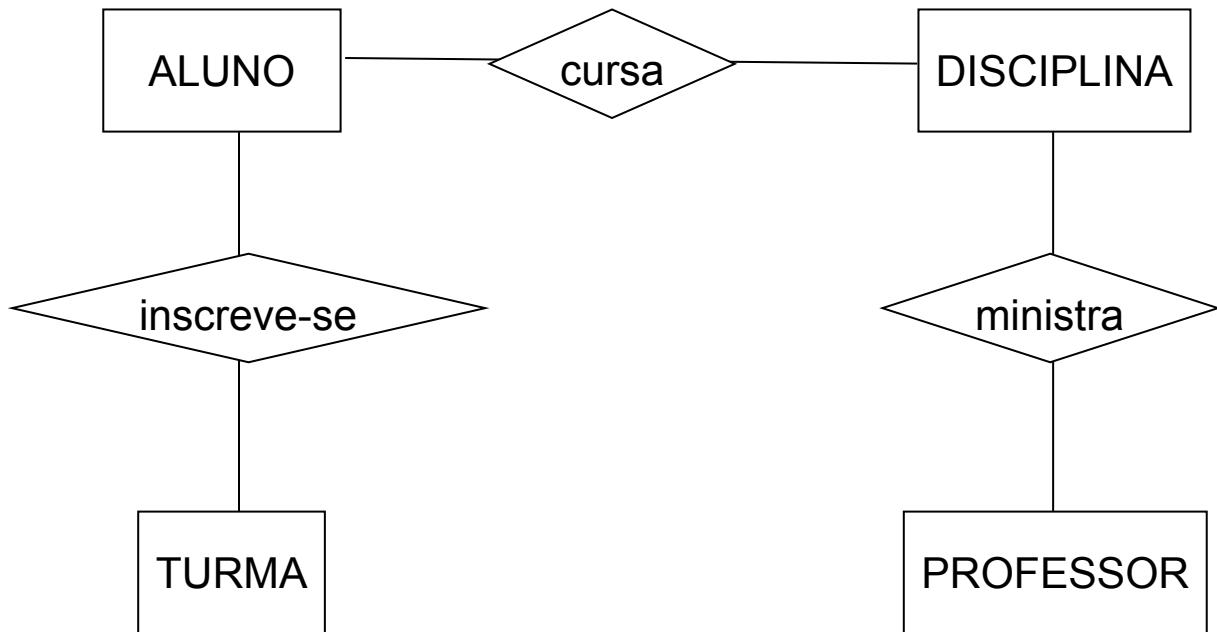
Cursa: sabemos que um aluno pode cursar várias disciplinas, bem como uma disciplina pode ser cursada por vários alunos. Isso está representado no diagrama a seguir pelo relacionamento CURSA, que é de muitos para muitos. Essa relação pode ser visualizada a partir do item 4 solicitado no minimundo.

Inscreve-se: note que o aluno se inscreve em uma turma, enquanto uma turma normalmente é composta por vários alunos. Isso está representado pelo relacionamento INSCREVE-SE, que é de muitos para um. Essa relação pode ser identificada no texto do minimundo no segundo parágrafo, quando o responsável pelo sistema diz que “é necessário informar em qual turma o aluno está inscrito”.

Ministra: Cada professor ministra uma ou mais disciplinas no colégio. Essa relação

pode ser percebida a partir do item 3 solicitado no minimundo.

Passo 3: Elaborar o DER mostrando entidades e relacionamentos



Fim da atividade

Prosseguindo esta aula, vamos estudar um pouco mais sobre entidades, atributos e relacionamentos.

Atributos de relacionamentos

Uma instância identifica individualmente uma entidade. Você pode observar, na Figura 2.12, que uma entidade possui várias instâncias e que cada instância está relacionada a uma entidade. Assim, uma entidade representa um conjunto de instâncias que interessam ao negócio.

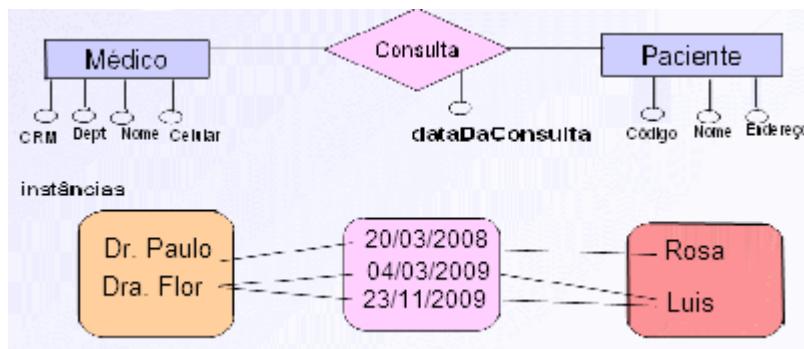


Figura 2.12. Atributos de relacionamento.

Neste exemplo observam-se duas entidades, chamadas Médico e Paciente; o nome do relacionamento é Consulta. A entidade Médico possui quatro atributos: CRM, Dept., Nome e Celular; a entidade Paciente possui três atributos, chamados Código, Nome e Endereço. A entidade Médico possui duas instâncias, Dr. Paulo e Dra. Flor. A entidade Paciente possui também duas instâncias: Rosa e Luis.

Também observa-se que o relacionamento chamado Consulta possui três instâncias e tem um único atributo, chamado dataDaConsulta.

Logo, o Médico realizou a consulta ao paciente em determinada data. O Dr. Paulo realizou a consulta à Paciente Rosa no dia 20/03/2008. A Dra. Flor realizou consultas em dois dias ao Paciente Luis (04/03/2009 e 23/11/2009).

Início da atividade online

Atividade 4 - Atende ao objetivo 2

Entre no ambiente virtual e resolva a atividade abaixo, enviando sua resposta ao tutor. Analise o minimundo a seguir e liste as entidades envolvidas e seus respectivos atributos. Em seguida, construa um diagrama ER apresentando os relacionamentos existentes entre as entidades que você encontrou. Repare que esse exercício de minimundo já apresenta atributos de relacionamento.

Minimundo 02

A locadora de vídeos Alugue Agora é uma loja que mantém filmes抗igos e novos para aluguel pelos seus clientes. Dada a crescente demanda por seus serviços, os donos da

loja decidiram contratar uma firma para criar um sistema com um banco de dados que permita identificar os filmes mais solicitados pelo público e os clientes que normalmente devolvem com atraso. Em conversa com os donos da loja, estes informaram que o banco de dados deve armazenar as seguintes informações sobre os filmes: título, atores principais e coadjuvantes, data de criação, país de origem, tipo e número de cópias.

Para cada filme, o sistema deve informar se possui uma cópia disponível para ser alugada. Eles também solicitaram que o sistema mantenha um cadastro dos clientes que normalmente alugam seus filmes. O sistema precisa manter também informações sobre cada filme alugado, ou seja: quem o alugou, quando e a data de devolução. A loja tem a política de que todos os filmes devem ser devolvidos no prazo de 3 dias. A locadora também possui um cadastro de fornecedores dos filmes disponíveis, que é composto por nome do fornecedor, endereço, telefone e data de cadastro.

O sistema também deverá ser capaz de emitir as seguintes listagens:

1. Relação dos clientes, contendo: código de inscrição na locadora, nome do cliente, data de nascimento, CPF, endereço e telefone;
2. Listagem dos filmes oferecidos em aluguel pela loja, apresentando para cada filme: código do filme, número de cópias, título, atores principal e coadjuvante, país de origem e tipo e número de vezes que ele foi alugado;
3. Relatório dos filmes alugados por cada cliente, exibindo: nome do cliente e nome do filme que ele alugou e se ele o devolveu na data correta;
4. O sistema deve ser capaz de emitir uma lista contendo os filmes que estão pendentes de devolução pelos clientes da loja e os dias de atraso, respectivamente.

Fim da atividade online

Conclusão

Como toda abstração, o projeto conceitual visa reduzir a complexidade da realidade para facilitar sua compreensão e representação. Assim, no nosso contexto, não teremos no projeto conceitual do banco de dados os dados que de fato serão armazenados. Teremos, sim, o formato dos dados que pretendemos armazenar.

Construir um modelo conceitual nem sempre é uma tarefa atrativa para um desenvolvedor. Entretanto, ao aplicar a metodologia sugerida nesta aula quanto à

identificação das entidades e relacionamentos, obteremos as características do banco de dados que irá armazenar as informações pertinentes ao propósito do mesmo.

O processo vivenciado durante a modelagem conceitual gera maior maturidade com relação aos objetivos do sistema, pois a etapa mais crítica é o momento em que é exigida ampla análise do contexto do problema apresentado na etapa de levantamento de informação. As análises das associações são exemplos claros da necessidade de muita atenção durante a criação do Modelo ER.

Resumo

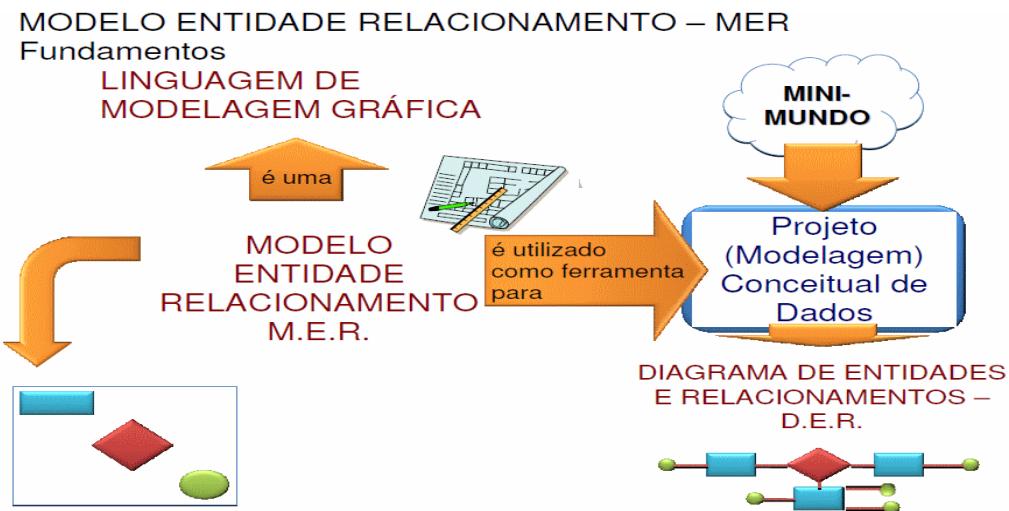
O projeto conceitual de banco de dados é feito pelo levantamento de requisitos, obtendo o modelo descritivo das necessidades dos usuários e a modelagem de dados usando o modelo ER (Entidade Relacionamento).

O modelo ER fornece as regras e conceitos para a criação do DER (diagrama Entidade Relacionamento), que deverá representar o banco de dados em questão em nível abstrato.

O modelo conceitual está formado pelo modelo descritivo e o modelo ER e representa as informações que existem no mundo real.

No modelo ER, uma entidade é um conjunto de objetos do mundo real sobre os quais se deseja manter a informação no banco de dados, sendo distinguível de outros objetos. A etapa dos relacionamentos entre entidades consiste em realizar uma análise das possíveis associações existentes entre as entidades alicerçadas na captura dos levantamentos dos requisitos.

Podemos concluir mostrando a seguinte figura:



Informações sobre a próxima aula

A transformação do projeto conceitual abstrato obtida nesta aula para o projeto lógico formalizando os conceitos do modelo ER será o tema da próxima aula.

O detalhamento e novas regras para a representação do DER também serão assuntos da Aula 3.

Referências bibliográficas

- DATE, C. J. *Introdução a sistemas de bancos de dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.
- CARVALHO, C. R. *SQL - Guia prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.
- ELMASRI R.; NAVATHE, S. *Sistemas de banco de dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.
- HEUSER, C. A. 2009. *Projeto de banco de dados*. 6^a ed. Porto Alegre: Bookman, 2009.
- SETZER, V. W.; CORRÊA DA SILVA, F. S. *Bancos de dados*. São Paulo: Edgard Blucher, 2005.
- SILBERSCHATZ, A.; KORTH, H. *Sistema de banco de dados*. 3^a ed. São Paulo: Pearson Makron Books, 2008.

Modelando e implementando bancos de dados relacionais
Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 3

Projeto conceitual – cardinalidade de relacionamentos e atributos

Meta

Apresentar conceito de cardinalidade de relacionamentos e atributos, importante para o detalhamento do projeto conceitual de banco de dados.

Objetivos

Ao final desta aula esperamos que você seja capaz de:

1. Descrever as cardinalidades para os relacionamentos;
2. Descrever as cardinalidades para os atributos de entidades.

Pré-requisitos

Para melhor aproveitamento desta aula, é importante você relembrar:

- os conceitos de banco de dados relacional e chave, vistos na Aula 1;
- os conceitos de projeto de banco de dados, projeto conceitual, levantamento de requisitos e regras de negócio, estudados na Aula 2.

Vamos contar?

Na última aula, você viu como construir modelos ER (entidade relacionamento) a partir da interpretação de minimundos, que são representações simplificadas de situações existentes no mundo real. A partir da análise dos minimundos podemos identificar a existência de entidades e relacionamentos.

Agora que você já tem tudo sobre entidades e relacionamentos na “ponta da língua”, podemos falar em outro conceito que está relacionado ao que já viu até agora...

Você já ouviu falar de números **cardinais**? É um conceito importante da matemática que também é muito usado na área de banco de dados, tanto para os relacionamentos

como para os atributos.

Início verbete

Cardinal

Indica o número ou quantidade dos elementos constituintes de um conjunto. Por exemplo: um conjunto de frutas é composto de: 1 maçã, 2 peras e 3 bananas. O cardinal corresponde ao número de elementos desse conjunto, que é igual a 6. É interessante destacar que se diferencia do ordinal porque o ordinal introduz ordem e dá idéia de hierarquia: primeiro, segundo, terceiro etc. O cardinal, por sua vez, nomeia o número de elementos constituintes, e esse é o nome do conjunto correspondente.

Fim do verbete

Imagine que em um determinado minimundo, tenhamos encontrado uma entidade chamada PESSOA. Ora, todos nós sabemos que pessoas costumam ter mais de um número de telefone. Normalmente, um telefone de casa e outros que são o número do celular e/ou do trabalho. Então, para o atributo telefone da entidade PESSOA, não temos apenas um único telefone, mas sim vários deles. Daí a ideia de cardinalidade, ou seja, mais de um elemento num determinado conjunto. Isso pode acontecer tanto para os atributos quanto para os relacionamentos. Nesta aula você verá, de forma mais detalhada, como isso funciona.



Figura 3.1. Números cardinais

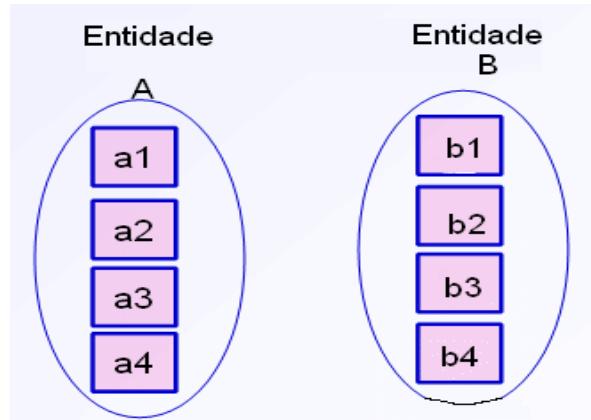
Fonte: <http://www.sxc.hu/photo/372094>

Autor: Tomasz A. Poszwa

Fim da introdução

Cardinalidade de relacionamentos

Uma propriedade importante dos relacionamentos é a especificação de quantas instâncias de uma entidade podem estar associadas a uma determinada instância de outra entidade.



A instância a1 da Entidade A está relacionada a quantas instâncias em B?

Existem 2 cardinalidades: máxima e mínima.

1º Caso: Relacionamento UM-PARA-UM ou 1:1

Uma instância da Entidade A está associada, no máximo, a uma instância de B e uma instância em B está associada, no máximo, a uma instância em A. Exemplo:

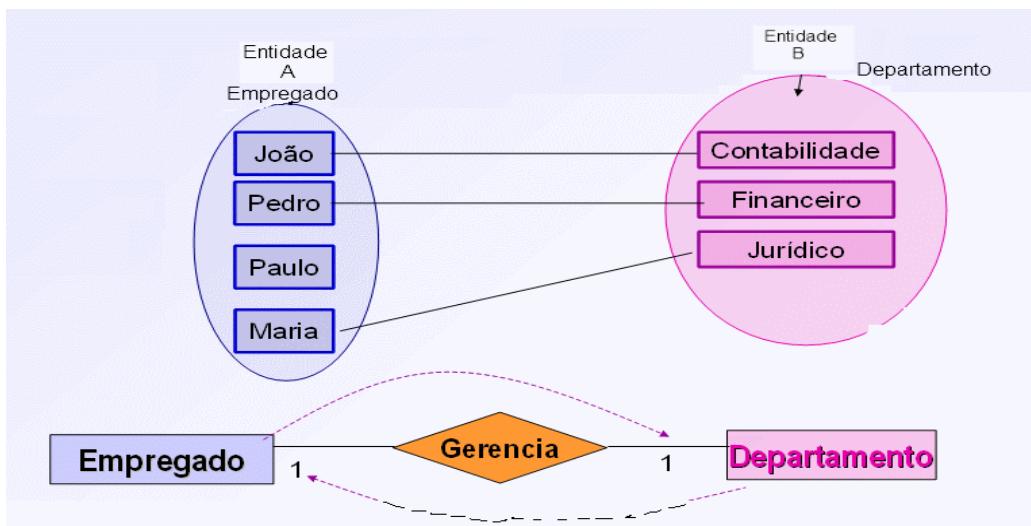


Figura 3.2. Relacionamento 1:1

Observe que, pelo diagrama de entidades, temos: um Empregado que gerencia, no

máximo, um Departamento; um Departamento é gerenciado por apenas um Empregado.

2º Caso: Relacionamento UM-PARA-MUITOS ou 1:N

Ocorre quando uma instância da Entidade A está associada a qualquer número de instâncias da Entidade B. Porém, uma instância da Entidade B está associada, no máximo, a uma instância de A. Exemplo:

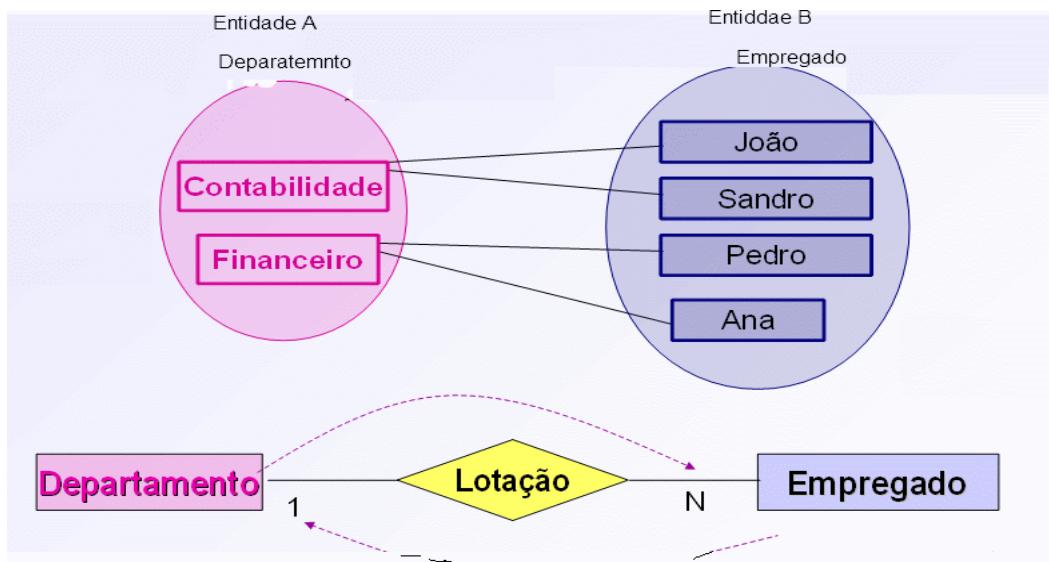


Figura 3.3. Relacionamento 1:N

Observe que, pelo diagrama de entidades, temos: um Departamento tem, lotados nele, N Empregados; um Empregado está lotado, no máximo, em um Departamento.

3º Caso: Relacionamento MUITOS-PARA-MUITOS ou M:N ou N:N

Ocorre quando uma instância da Entidade A está associada a qualquer número de instâncias da Entidade B e uma instância de B está associada a qualquer número de instâncias de A. Exemplo:

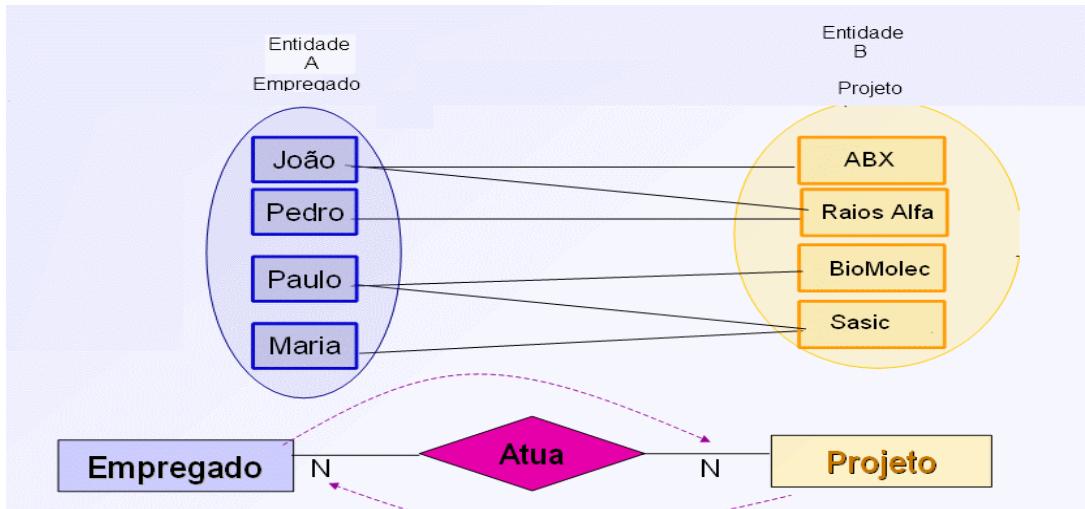


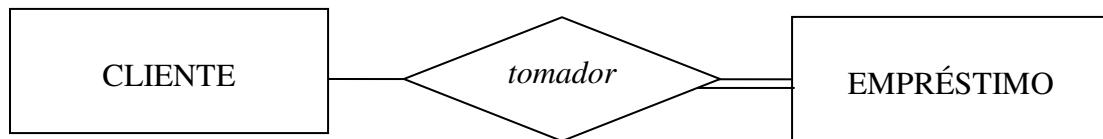
Figura 3.4. Relacionamento N:N

Observe que, pelo diagrama de entidades, temos: um Empregado atua em N Projetos e um Projeto tem atuação de N Empregados.

Participação total e parcial

Para analisar o grau de participação das entidades dentro de um relacionamento, precisamos verificar o relacionamento entre as entidades partindo de ambos os sentidos, ou seja, de cada uma das entidades envolvidas no relacionamento analisado.

Exemplo:



Participação total (indicada por linha dupla)

Ocorre quando cada instância de uma determinada entidade participa de um relacionamento.

No exemplo acima, a participação de EMPRÉSTIMO em *tomador* é total, visto que cada empréstimo precisa ter um *cliente* associado através de *tomador*.

Participação parcial

Ocorre quando nem todas as instâncias de uma determinada entidade participam de um relacionamento.

No exemplo acima, a participação de CLIENTE em *tomador* é parcial.

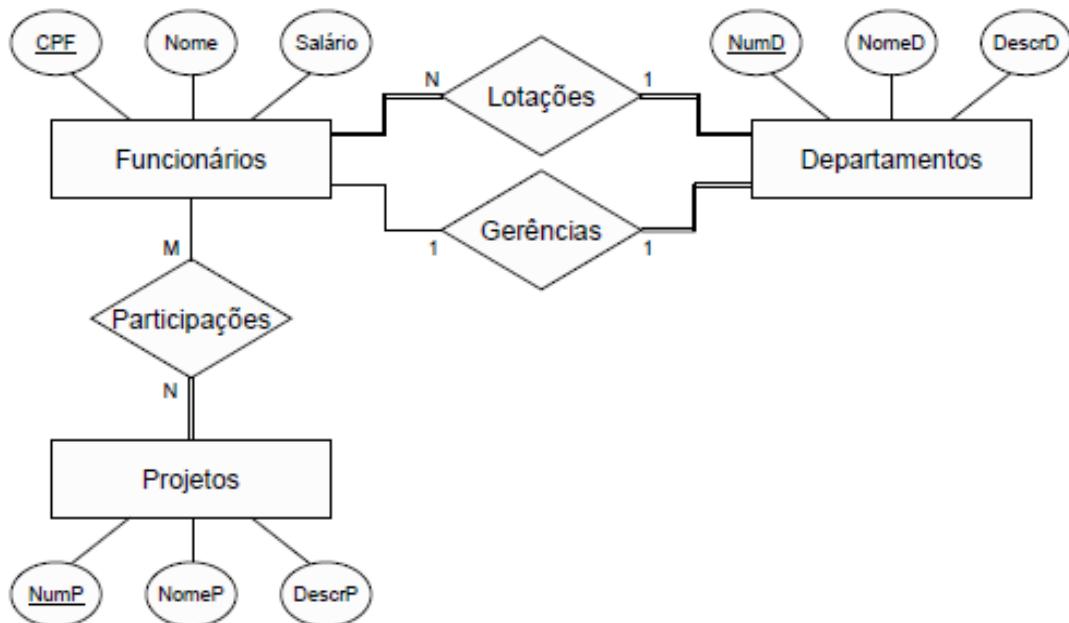


Figura 3.5. Relacionamentos parcial e total

Analisando os relacionamentos da Figura 3.5 temos que:

- ✓ O conjunto de relacionamento Gerências é total em Departamentos, pois todo departamento precisa ter um gerente, e parcial em Funcionários, pois nem todo funcionário é gerente de departamento;
- ✓ O conjunto de relacionamentos Lotações é total em Funcionários e total em Departamentos.
- ✓ O conjunto de relacionamentos Participações é parcial em Funcionários, pois nem todo funcionário participa em projetos, e total em Projetos, pois todo projeto precisa ter um funcionário.

Logo, o Modelo ER permite expressar cardinalidades mínimas e máximas em cada

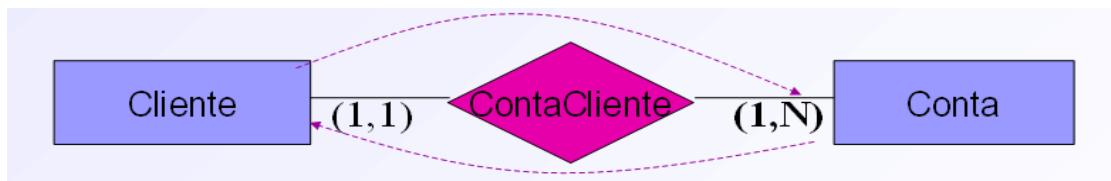
relacionamento. As cardinalidades possíveis são: (1, 1), (1, N), (0, 1), (0,N), (N, N).

Cardinalidade mínima = 1 (relacionamento obrigatório)

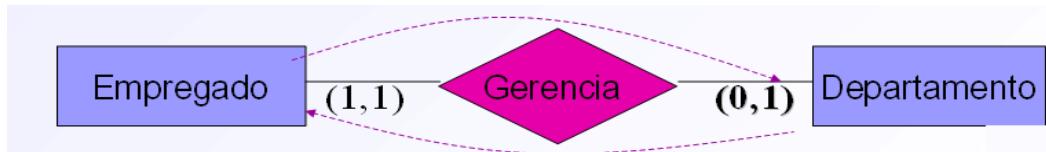
Cardinalidade mínima = 0 (relacionamento opcional)

Exemplo: Relacionamento obrigatório

- ✓ Cada instância de cliente paga no mínimo uma conta e no máximo N contas.
- ✓ Cada instância de conta pode ser paga, no mínimo, por um cliente e, no máximo, por um cliente.



- ✓ Cada instância de Empregado gerencia, no mínimo, 0 e, no máximo, um departamento;
- ✓ Cada instância de Departamento é gerenciada por, no mínimo, um empregado e, no máximo, um empregado.



Ou

- ✓ Um genitor pode ter de 1 a N filhos.
- ✓ Um filho tem, no mínimo, dois e, no máximo, dois genitores.



Início da atividade

Atividade 1 - Atende ao objetivo 1

Analise o minimundo a seguir e liste as entidades envolvidas e seus respectivos atributos. Em seguida, descreva as cardinalidades para os relacionamentos apresentados. Por fim, construa um diagrama ER apresentando os relacionamentos

existentes entre as entidades que você encontrou.

Minimundo 01

A empresa AutoRun é especializada no conserto de veículos automotores, incluindo carros, caminhões e motos. Devido a recentes prejuízos com o desaparecimento de peças na empresa, o gerente decidiu implantar um sistema para controlar os orçamentos emitidos e as peças utilizadas nos respectivos serviços prestados pela empresa.

O gerente informou que, tanto no caso dos orçamentos quanto para os serviços realizados, normalmente o mecânico responsável anota o número do orçamento ou serviço, o valor total correspondente, a data da sua realização e acrescenta uma descrição detalhada do que foi orçado ou realizado. Ele também precisa relacionar todas as peças que foram previstas no orçamento ou utilizadas na execução do serviço, se for o caso.

No caso da solicitação de um orçamento, as seguintes informações do cliente também são registradas pelo mecânico: CPF, nome do cliente, endereço e telefone para contato. No caso de um serviço, também são anotados o número do orçamento correspondente (se houver) e os nomes de todos os mecânicos que o executaram. Ao final do serviço, é emitida uma nota fiscal que deverá ser paga pelo cliente.

O gerente informou ser imprescindível que o sistema armazene tanto o número da nota fiscal emitida para pagamento quanto a data em que ela foi paga pelo cliente. Além de controlar os orçamentos e serviços realizados pela empresa, o sistema deverá ser capaz de emitir os seguintes relatórios:

1. Listagem de peças em estoque na empresa, contendo o código da peça, o nome, o fabricante, o valor unitário e a quantidade disponível;
2. Relação mensal de gastos com os mecânicos, mostrando matrícula, nome, telefone e salário de cada um deles;
3. Listagem completa dos serviços realizados por cada mecânico no mês, exibindo o nome e a matrícula do mecânico, bem como o número e o valor dos serviços executados.

Resposta comentada

Após analisarmos o minimundo, efetuamos os seguintes passos:

Passo 1: Identificar as entidades com seus respectivos atributos

Entidade Cliente: Repare que o cliente é quem inicia todo o fluxo dos serviços prestados e paga por eles. Sua importância pode ser visualizada ao final do segundo parágrafo do texto do minimundo. Seus atributos são: CPF, nome do cliente, endereço e telefone.

Entidade Orçamento: Repare que orçamento possui várias informações relevantes para o sistema, como pode ser visto na parte inicial do segundo parágrafo do minimundo. Seus atributos são: número de identificação, valor, data e descrição.

Entidade Serviço: Repare que orçamento possui várias informações relevantes para o sistema, como pode ser visto na parte inicial do segundo parágrafo do minimundo. Seus atributos são: número de identificação, valor, data e descrição.

Entidade Mecânico: Repare que o mecânico possui várias informações relevantes para o sistema, como pode ser visto no item 2 das solicitações do minimundo. Seus atributos são: matrícula, nome, telefone e salário.

Entidade Peça: Repare que peça possui várias informações relevantes para o sistema, como pode ser visto no item 1 das solicitações do minimundo. Seus atributos são: código, nome, quantidade, valor unitário e fabricante.

Passo 2: Identificar os relacionamentos entre as entidades e seus respectivos atributos

Relacionamento Solicita: Note que o fluxo de trabalho se inicia quando o cliente solicita um orçamento à oficina AutoRun para o reparo do seu veículo. Isso pode ser visualizado através do texto presente no parágrafo dois: 'No caso da solicitação de um orçamento, as seguintes informações do cliente também são registradas pelo mecânico'.

Relacionamento É-elaborado-Por: Para cada solicitação de orçamento há um mecânico responsável por elaborá-lo. Isso pode ser visualizado no texto presente no parágrafo dois: “tanto no caso dos orçamentos quanto para os serviços realizados, normalmente o mecânico responsável toma nota do número do orçamento ou serviço”.

Relacionamento Produz: Para cada serviço normalmente há um orçamento correspondente, apesar de não ser obrigatório. Isto fica evidenciado pelo texto “No caso de um serviço, também são anotados o número do orçamento correspondente (se houver)”, presente ao final do parágrafo dois.

Relacionamento Executa: O serviço orçado normalmente é executado por um ou mais mecânicos. Isso pode ser observado através do texto: “No caso de um serviço, também são anotados o número do orçamento correspondente (se houver) e os nomes de todos os mecânicos que o executaram”, presente ao final do parágrafo dois.

Relacionamento Lista: Além dos atributos do orçamento, é necessário listar as peças previstas que estarão envolvidas no reparo do veículo. Isso é evidenciado pelo texto “Ele também precisa relacionar todas as peças que foram previstas no orçamento ou utilizadas na execução do serviço, se for o caso”, presente no parágrafo dois.

Relacionamento Usa: Além dos atributos do serviço, é necessário listar as peças utilizadas no reparo do veículo. Isso é evidenciado pelo texto “Ele também precisa relacionar todas as peças que foram previstas no orçamento ou utilizadas na execução do serviço, se for o caso”, presente no parágrafo dois.

Relacionamento Paga: Após a prestação do serviço, o cliente paga uma nota fiscal no valor do serviço prestado. Esse relacionamento possui dois atributos, que são: número da nota fiscal e data de pagamento. Isso fica evidenciado pelo texto: “Ao final do serviço, é emitida uma nota fiscal que deverá ser paga pelo cliente. O gerente informou ser imprescindível que o sistema armazene tanto o número da nota fiscal emitida para

pagamento quanto a data em que esta foi paga pelo cliente”, encontrado no terceiro parágrafo do minimundo.

Passo 3: Identificar as cardinalidades mínima e máxima de cada relacionamento

Relacionamento Solicita: Cada cliente pode solicitar 1 ou mais orçamentos; daí temos a cardinalidade (1,N); por outro lado, cada orçamento está associado a um único cliente; portanto, a cardinalidade será de (1,1).

Relacionamento É-elaborado-Por: Cada orçamento solicitado será elaborado por um único mecânico responsável. Portanto, o relacionamento será de (1,1). Por outro lado, um determinado mecânico pode não ter elaborado nenhum ou já ter feito vários orçamentos, daí a cardinalidade será de (0,N).

Relacionamento Produz: Cada orçamento produz um serviço quando o cliente o aceita ou nenhum serviço quando o cliente não concorda com o mesmo. Daí a cardinalidade será de (0,1). Por outro lado, cada serviço pode ter sido oriundo de um orçamento ou de nenhum orçamento, quando o cliente solicita o serviço sem querer saber de antemão o valor do reparo. Por isto, a cardinalidade será de (0,1).

Relacionamento Executa: Cada serviço pode ser executado por um ou mais mecânicos, daí a cardinalidade ser de 1,N. Por outro lado, cada mecânico pode não estar alocado à execução de um serviço ou estar executando ou ter executado vários serviços. Portanto, a cardinalidade será de 0,N.

Relacionamento Lista: Cada orçamento pode requerer o uso de várias peças para o reparo do veículo ou nenhuma, no caso de o serviço residir apenas em prestação de mão de obra, sem substituição de peças. Assim, a cardinalidade é de 0,N. Em contrapartida, uma peça pode ser listada por vários orçamentos ou mesmo nenhum, gerando a cardinalidade 0,N.

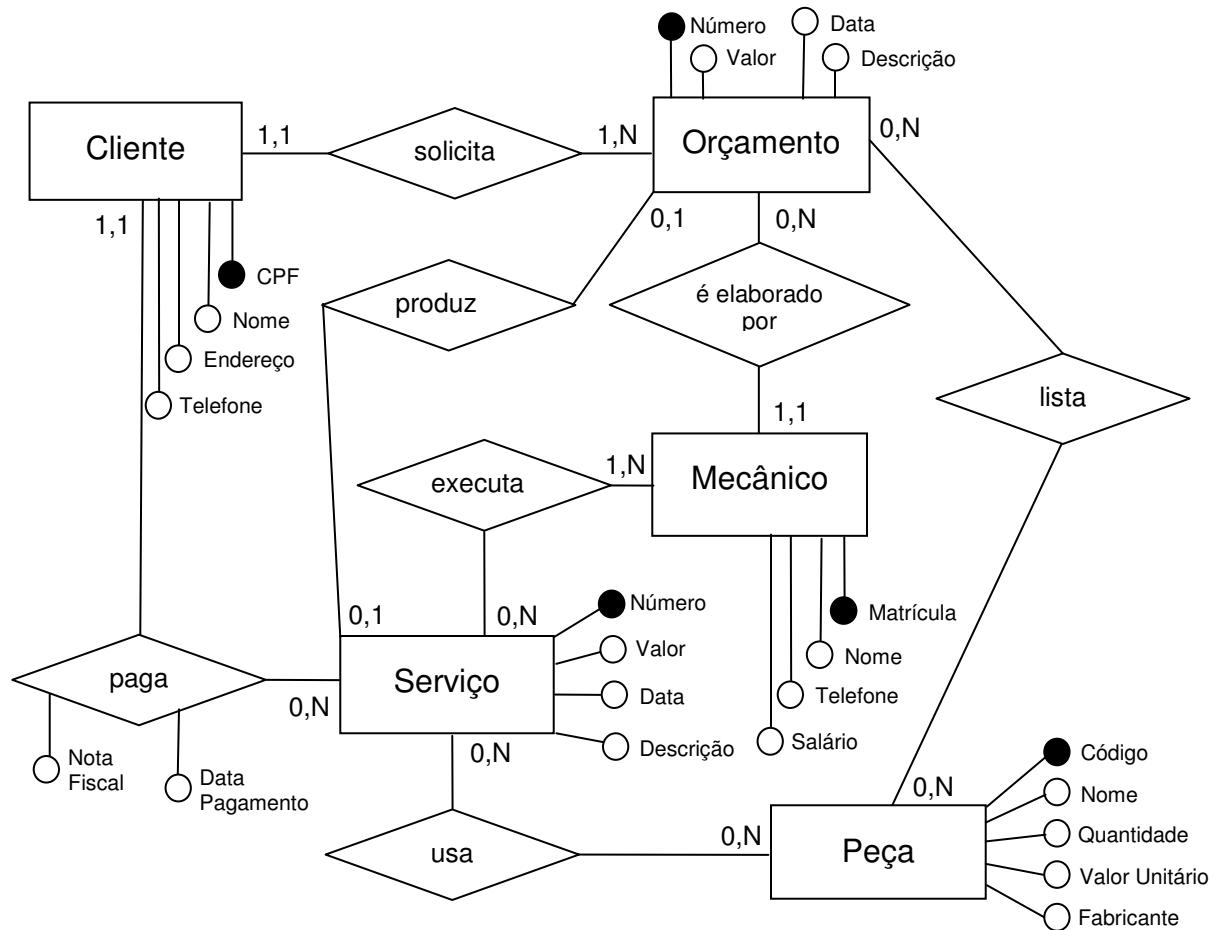
Relacionamento Usa: Cada serviço pode usar nenhuma ou várias peças. Assim, a

cardinalidade será de 0,N. Por outro lado, uma peça pode não ser usada em nenhum serviço ou em vários serviços, daí a cardinalidade é de 0,N.

Relacionamento Paga: Cada serviço prestado será pago por um, e somente um, cliente, gerando a cardinalidade 1,1. Em contrapartida, um cliente pode pagar vários serviços ou se recusar a pagar algum serviço executado, gerando a cardinalidade 0,N.

Passo 4: Elaborar o DER

O DER representando o minimundo descrito deve ser elaborado considerando as entidades, seus respectivos atributos, o relacionamento entre elas, com os atributos desses relacionamentos e, finalmente, a cardinalidade de cada relacionamento. O DER deve, então, refletir todas as informações relevantes que foram capturadas a partir da descrição do mundo real.



Fim da atividade

Início da atividade online

Atividade 2 - Atende ao objetivo 1

Acesse o ambiente virtual e resolva a atividade a seguir, enviando sua resposta ao tutor.

Minimundo 02

O Hospital Cura-te é uma instituição de renome. Uma recente auditoria detectou que remédios inadequados foram administrados aos seus pacientes, colocando o hospital numa situação constrangedora. Frente ao ocorrido, a diretoria percebeu a necessidade de implantar um novo sistema, de modo a poder controlar com mais precisão o

prontuário de cada um dos seus pacientes.

Quando o hospital dá entrada a um novo paciente, é realizado um diagnóstico inicial e é preenchida uma ficha com os seguintes dados: nome, RG, CPF, endereço e telefone. Também são anotados o médico responsável e a situação geral do paciente.

Durante a sua estada no hospital, o paciente recebe um número de registro, que é utilizado para o acompanhamento de tudo o que for realizado, mediante o preenchimento de registros do seu prontuário médico.

Além de controlar a entrada e a saída dos pacientes do hospital e os remédios ministrados a eles, o sistema também precisa produzir os seguintes relatórios:

1. Listagem de médicos de plantão para os fins de semana, incluindo nome, especialidade, RG, CPF, CRM, endereço e telefone;
2. Emissão do prontuário completo do paciente, incluindo seus dados cadastrais bem como todas as ocorrências, exames, consultas e medicamentos ministrados a ele. Cada ocorrência do prontuário deve possuir data, o médico responsável, uma descrição e um tipo (exame, consulta ou medicação). No caso de uma medicação, devem ser relacionados todos os remédios que foram administrados ao paciente (código, nome e fabricante), bem como as dosagens praticadas.

Analise o minimundo apresentado e liste as entidades envolvidas e seus respectivos atributos. Em seguida, descreva as cardinalidades existentes nos relacionamentos. Por fim, construa um diagrama ER apresentando os relacionamentos e cardinalidades existentes entre as entidades que você encontrou.

Fim da Atividade Online

Cardinalidades de atributos

Atributo monovalorado ou atômico

Possui um valor único em uma entidade ou assume um único valor em um certo instante de tempo.

Exemplo: Atributo monovalorado ==> nome

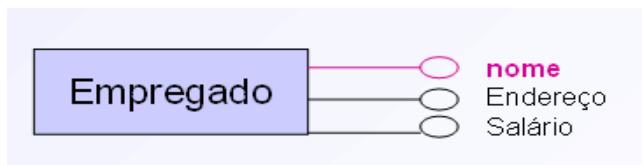


Figura 3.6. Entidade - atributo atômico

Atributo multivalorado

Possui mais de um valor para cada instância de entidade.

Exemplo: Atributo multivalorado ==> telefone

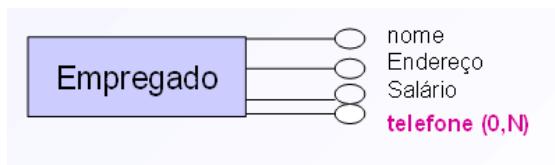


Figura 3.7. Entidade - atributo multivalorado

Veja mais um exemplo para deixar isso claro. Observe a figura a seguir:

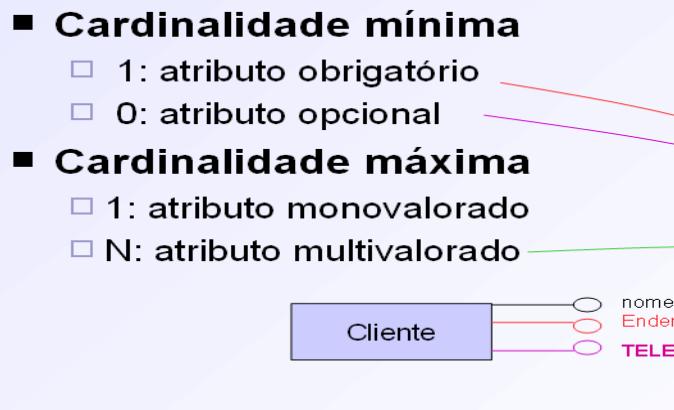


Figura 3.8. Entidade - atributo - cardinalidade mínima e máxima

Portanto, o atributo endereço da Entidade Cliente possui uma cardinalidade mínima obrigatória e uma cardinalidade máxima multivalorada, ou seja, uma instância de Cliente pode ter um ou mais endereços. Por outro lado, uma instância de Cliente pode ter nenhum ou vários telefones.

As Regras de Negócio aparecem nas cardinalidades mínimas e máximas da seguinte

forma:



Figura 3.9. Atributo - regras de negócio - cardinalidade mínima e máxima

Podemos também observar as classes de relacionamentos de cardinalidade:

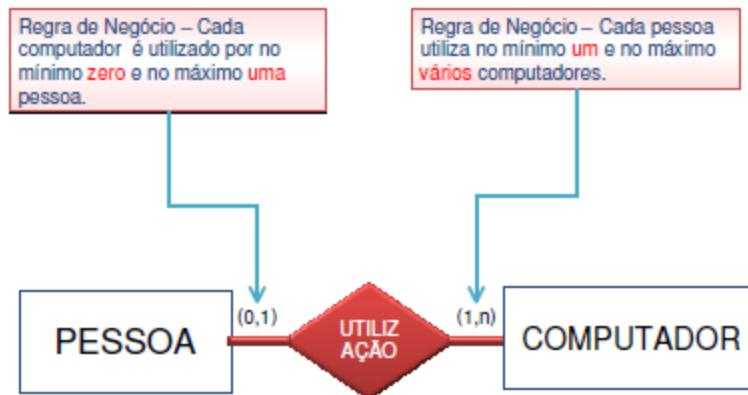


Figura 3.10. Classes de relacionamento de cardinalidade

Início da atividade

Atividade 3 - Atende aos objetivos 1 e 2

Analise o minimundo a seguir e liste as entidades envolvidas e seus respectivos atributos. Em seguida, descreva as cardinalidades relacionadas aos relacionamentos e atributos. Por fim, construa um diagrama ER apresentando os relacionamentos existentes entre as entidades que você encontrou.

Minimundo 03

A empresa Importex é especializada na importação e venda local de produtos originados de outros países.

A empresa tem perdido várias vendas devido à falta do produto em estoque quando o cliente faz um pedido para comprá-lo. A diretoria da empresa decidiu, então, implantar um sistema que permita informar o nível de estoque dos produtos e monitorar as vendas para evitar novos prejuízos.

O usuário designado pela diretoria informou que, quando o cliente solicita um ou mais produtos, o funcionário verifica se os mesmos existem em estoque. Em caso negativo, ele anota os produtos que devem ser comprados e sua respectiva quantidade de reposição. Se os produtos estiverem disponíveis, o atendente encaminha ao setor de vendas um pedido especificando os produtos e quantidades solicitadas pelo cliente. Todo pedido possui um número único que o identifica. Os produtos são, então, separados e embalados. Simultaneamente, o setor de vendas emite uma nota fiscal que relaciona os produtos adquiridos, as respectivas quantidades e preços de venda. O cliente efetua o pagamento no valor total da nota fiscal. Clientes com cartão de fidelidade têm direito a 10% de desconto sobre o total da nota.

O sistema a ser desenvolvido deve registrar cada venda efetuada, informando o número da venda (originado do pedido), o número da nota fiscal, a data e o valor total. A fim de saber se o cliente tem direito a desconto, a empresa mantém um cadastro de clientes onde constam CPF, endereço, nome, número do cartão de fidelidade e seus telefones. Da mesma forma, um cadastro de fornecedores lista todas as informações necessárias para contato no caso de aquisição de novos produtos.

Com o objetivo de evitar que novas vendas sejam perdidas devido à falta de produto em estoque, o sistema deverá ser capaz de emitir os seguintes relatórios:

- a. Listagem de pedidos pendentes, relacionando para cada um o seu número, data e valor;
- b. Listagem de produtos sem estoque disponível, contendo código do produto, sua

- descrição e valor unitário;
- c. Relatório das vendas diárias dos produtos, relacionando o número da nota fiscal, código, descrição do produto, data e valor da nota fiscal;
 - d. Listagem dos fornecedores de produtos ordenada por país de origem. A listagem deve exibir o código, a razão social, o CNPJ, o(s) endereço(s) e o(s) telefone(s) para contato.

Diagramação favor deixar um espaço equivalente a 20 linhas para a resposta.

Resposta comentada

Após analisar o minimundo, efetuamos os seguintes passos:

Passo 1: Identificar as entidades com seus respectivos atributos

Entidade Cliente: Repare que o cliente é quem inicia todo o fluxo da compra dos produtos e paga por ela. Sua importância pode ser visualizada em vários parágrafos do minimundo, mas, principalmente, ao final do parágrafo 3 (cartão fidelidade). Seus atributos são: CPF, nome do cliente, endereço, telefone(s) – atributo multivvalorado (cardinalidade maior que 1) – e cartão de fidelidade.

Entidade Pedido: O pedido resulta da decisão do cliente de adquirir certo produto. Repare que pedido possui várias informações relevantes para o sistema, como pode ser visto no parágrafo 3 do minimundo e no item a do parágrafo 5. Seus atributos são: Número de identificação, data e valor.

Entidade Produto: Os produtos são alocados aos pedidos e fornecidos pelos fornecedores. Aparecem em vários locais no texto do minimundo. Nitidamente, tornam-se mais importantes por conta da solicitação b de relatório para o sistema, que aparece no parágrafo 5. Seus atributos são: código, descrição, valor unitário e estoque.

Entidade Venda: A venda aparece como a concretização do pedido realizado pelo cliente. É uma entidade que se torna importante em virtude da solicitação c de relatório para o sistema que aparece no parágrafo 5. Seus atributos são: número, data, valor e

nota fiscal.

Entidade Fornecedor: O fornecedor é aquele que provê os produtos para a importadora. Torna-se relevante como entidade devido à solicitação de relatório para o sistema que aparece no parágrafo 5. Seus atributos são: Código, Razão Social, CNPJ, país de origem, endereço(s) e telefone(s). Estes dois últimos são atributos multivvalorados, podendo conter mais de uma informação (cardinalidade maior que 1).

Passo 2: Identificar os relacionamentos entre as entidades e seus respectivos atributos

Relacionamento Realiza: É o relacionamento entre as entidades Cliente e Pedido. Note que o fluxo de trabalho se inicia quando o cliente faz um pedido à importadora: “quando o cliente solicita um ou mais produtos, o funcionário verifica se os mesmos existem em estoque... Se os produtos estiverem disponíveis, o atendente encaminha o pedido ao setor de vendas, especificando os produtos e as quantidades solicitadas pelo cliente”.

Relacionamento Relaciona: É o relacionamento entre as entidades Pedido e Produto. Para cada pedido do cliente, haverá produtos envolvidos. Isso pode ser visualizado através de “quando o cliente solicita um ou mais produtos, o funcionário verifica se os mesmos existem em estoque... Se os produtos estiverem disponíveis, o atendente encaminha o pedido ao setor de vendas, especificando os produtos”.

Relacionamento Produz: É o relacionamento entre as entidades Pedido e Venda. Cada pedido realizado pelo cliente gera uma venda. Isso ficada evidenciado pelo texto “Todo pedido possui um número único que o identifica. Os produtos são, então, separados e embalados. Simultaneamente, o setor de vendas emite uma nota fiscal”.

Relacionamento É-fornecido-Por: É o relacionamento entre as entidades Produto e Fornecedor. Os produtos vendidos pela empresa no mercado local advêm de fornecedores no exterior. Isso fica evidenciado pelo texto “um cadastro de fornecedores lista todas as informações necessárias para contato no caso de aquisição de novos

produtos" e pelo item d do parágrafo 5.

Passo 3: Identificar as cardinalidades mínima e máxima de cada relacionamento

Relacionamento Realiza: Cada cliente pode solicitar nenhum ou vários produtos, daí temos a cardinalidade (0,N). Por outro lado, um pedido, para existir, tem que ter sido realizado por um cliente e, ao mesmo tempo, só pode estar associado a um único cliente. Daí a cardinalidade é de (1,1).

Relacionamento Relaciona: Cada pedido relaciona pelo menos um produto ou vários produtos. Daí a cardinalidade ser (1,N). Já um produto pode estar relacionado por nenhum pedido (caso nenhum cliente tenha desejado comprá-lo) ou vários pedidos. Portanto, a cardinalidade é (0,N).

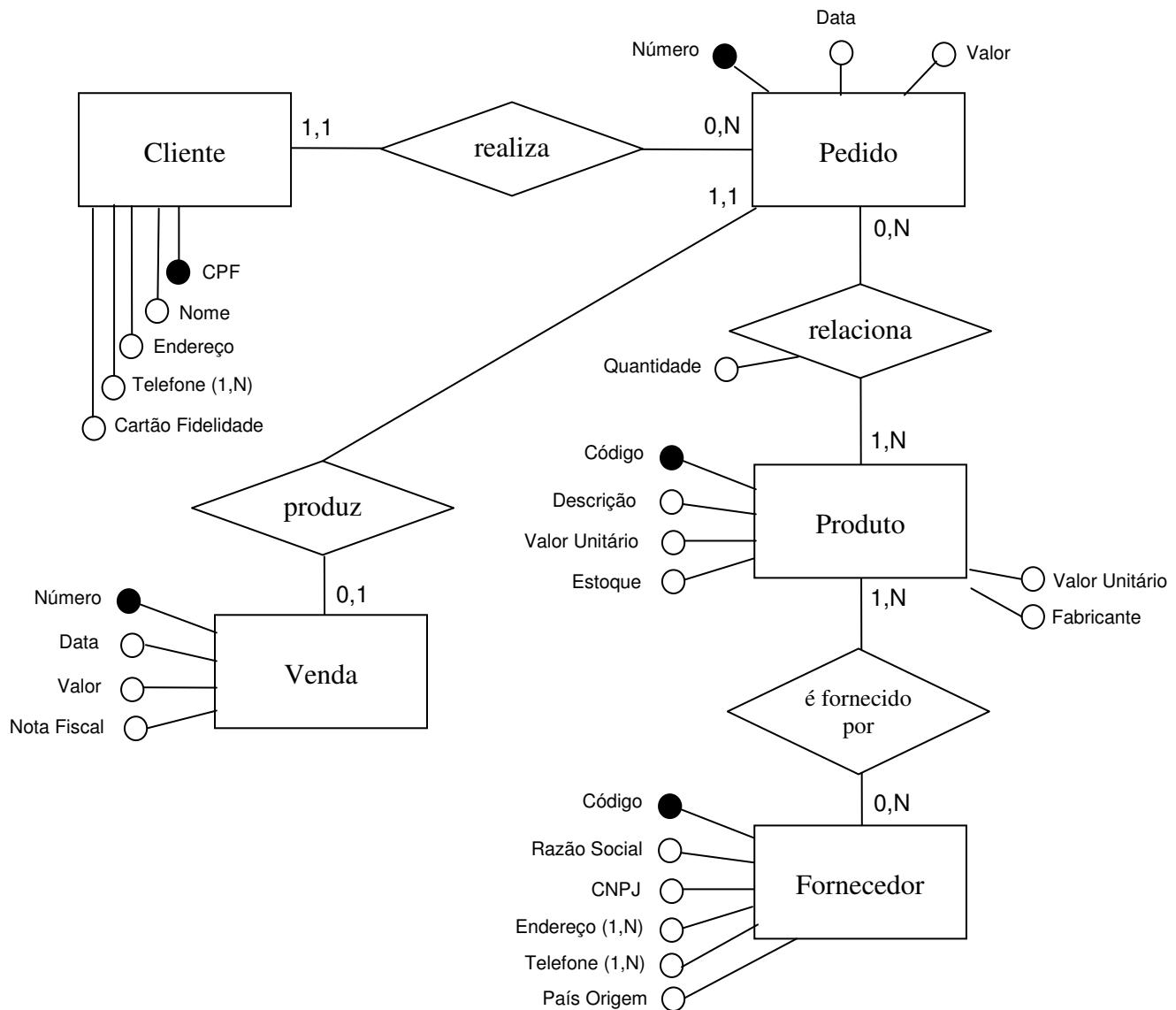
Relacionamento Produz: Cada pedido realizado pelo cliente produz nenhuma venda (caso o produto não exista no estoque) ou no máximo uma venda, já que estará associado a uma nota fiscal de venda. A cardinalidade será de (0,N). Por outro lado, uma venda está necessariamente associada a pelo menos um pedido de cliente (já que este é o gerador da venda) e no máximo a 1. Isso resulta numa cardinalidade (1,1).

Relacionamento É-fornecido-Por: Cada produto pode ser fornecido por 1 ou mais fornecedores. Entretanto, pode acontecer de um único fornecedor deixar de fornecer um determinado produto ao longo do tempo. Portanto, a cardinalidade será de (0,N). Em contrapartida, um fornecedor tem que fornecer ao menos um produto para permanecer ativo no cadastro do sistema. Ele também pode fornecer vários produtos que a empresa importa. Assim, a cardinalidade será de (1,N).

Passo 4: Elaborar o DER

O DER representando o minimundo descrito deve ser elaborado considerando as entidades, seus respectivos atributos, o relacionamento entre elas, os atributos desses relacionamentos; finalmente, deve expressar a cardinalidade de cada relacionamento. O DER deve, então, refletir todas as informações relevantes que foram capturadas a

partir da descrição do mundo real, incluindo os atributos multivvalorados das entidades (cardinalidades dos atributos).



Fim da atividade

Início da atividade

Atividade 4 - Atende aos objetivos 1 e 2

Entre no ambiente virtual e resolva a atividade a seguir, enviando sua resposta ao tutor. Analise o minimundo a seguir e liste as entidades, atributos, relacionamentos e cardinalidades dos relacionamentos e dos atributos. Em seguida, construa um

diagrama ER.

Minimundo 04

Uma fábrica de brinquedos exclusivos (cada modelo é único e projetado por *designers* famosos) deseja um sistema para controlar a sua produção. A fábrica conta atualmente com 1.700 funcionários; a maior parte deles é de montadores, trabalhando na atividade fim. A fábrica possui aproximadamente 500 máquinas de montagem de diversos tipos e de diversos fabricantes.

Para ingressar como montador, o funcionário é avaliado para determinar em que tipo de máquina ele possui habilitação. Cada máquina pode realizar um ou mais tipos de montagem.

Cada brinquedo é produzido integralmente por um montador em uma máquina; nesse período, nem o montador nem a máquina podem ser alocados para outra coisa.

A remuneração dos montadores é semanal. Baseia-se numa alíquota fixa (10%) sobre o preço de venda de cada brinquedo. Nenhum montador pode receber menos que um determinado valor mínimo, que é negociado no momento da contratação de cada um. Os montadores são divididos em equipes, cada uma possuindo um supervisor, que é o responsável pela qualidade do que é produzido e pela monitoração das máquinas que estão em conserto. A máquina só vai para o conserto após o término da produção do brinquedo.

A fábrica necessita das seguintes informações:

- a. Relatório de brinquedos produzidos por um montador em um determinado período no seguinte formato: modelo do brinquedo, descrição do modelo, data e hora de início e término da fabricação, código da máquina de montagem, localização da máquina e fabricante;
- b. Relatório dos montadores sem produção no período (matrícula do montador, nome, valor mínimo negociado);

- c. Relação das máquinas disponíveis, informando para cada uma o seu fabricante e o(s) tipo(s) de montagem que aceita;
- d. Quais montadores estão disponíveis e habilitados a trabalhar em um tipo de máquina no momento;
- e. Relação das máquinas que estiveram mais de sete vezes em conserto, contendo: código da máquina e, para cada conserto, matrícula e nome do supervisor responsável, data início e término do conserto.

Fim da atividade online

Conclusão

Após uma análise mais detalhada dos minimundos e dos exemplos aqui colocados, podemos perceber que os relacionamentos incluem um número mínimo e um número máximo de ocorrências quando associados às entidades que os compõem. Observamos também que essas entidades podem participar de forma parcial ou total em um relacionamento. Além disso, há diversos casos em que atributos comuns, tais como endereço e telefone, podem ocorrer mais de uma vez dentro de uma entidade específica, como PESSOA, por exemplo. Dessa forma, chegamos a um nível maior de definição do modelo conceitual que agora é capaz de refletir de forma um pouco mais apurada a realidade descrita dentro dos minimundos.

Resumo

A cardinalidade em relacionamentos indica quantas instâncias de uma entidade podem estar associadas a uma determinada instância de outra entidade.

Para gerar a cardinalidade, devemos perguntar: quantas instâncias no mínimo e no máximo poderão existir entre a entidade A e a entidade B em uma associação? Temos os seguintes relacionamentos:

- a) Relacionamento UM-PARA-UM ou 1:1

Uma instância da Entidade A está associada, no máximo, a uma instância de B e uma instância em B está associada, no máximo, a uma instância em A.

- b) Relacionamento UM-PARA-MUITOS ou 1:N

Ocorre quando uma instância da Entidade A está associada a qualquer número de

instâncias da Entidade B, porém uma instância da Entidade B está associada, no máximo, a uma instância de A.

c) Relacionamento MUITOS-PARA-MUITOS ou M:N ou N:N

Ocorre quando uma instância da Entidade A está associada a qualquer número de instâncias da Entidade de B e uma instância de B está associada a qualquer número de instâncias de A.

O Modelo ER permite expressar cardinalidades mínimas e máximas em cada relacionamento. As cardinalidades possíveis são: (1, 1), (1, N), (0,1), (0,N), (N, N).

Cardinalidade mínima = 1 (relacionamento obrigatório).

Cardinalidade mínima = 0 (relacionamento opcional).

Os relacionamentos também podem ter o grau de participação das entidades envolvidas. Uma participação total ocorre quando cada instância de uma determinada entidade participa de um relacionamento. Em uma participação parcial nem todas as instâncias de uma determinada entidade participam de um relacionamento.

Os atributos também podem ter cardinalidade, ou seja, atributo monovalorado (possui um valor único em uma entidade) e atributo multivlorado (possui mais de um valor para cada instância de entidade).

Informações sobre a próxima aula

Os relacionamentos podem envolver uma quantidade de entidades diferente de dois (únário, ternário e outros). Quando um relacionamento envolve apenas uma única entidade, diz-se unário. Quando envolve três entidades é ternário, e assim por diante. Também serão abordados casos especiais de relacionamentos, que são as especializações e generalizações. Uma análise mais detalhada das condições presentes nos minimundos mostra ainda a existência das restrições de integridade. Todos esses temas serão abordados na próxima aula, finalizando os conceitos necessários para a construção de um modelo ER completo.

Referências bibliográficas

DATE, C. J. *Introdução a sistemas de bancos de dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.

CARVALHO, C. R. *SQL - Guia prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.

ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.

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

SETZER, V. W. & CORRÊA DA SILVA, F. S. *Bancos de dados*. São Paulo: Edgard Blucher, 2005.

SILBERSCHATZ, A.; KORTH, H. *Sistema de banco de dados*. 3^a ed. São Paulo: Pearson Makron Books, 2008.

Modelando e Implementando Bancos de Dados Relacionais
Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 4: Modelagem conceitual: tipos de relacionamentos, generalizações, especializações de entidades e restrições de integridade

Meta

Apresentar conceitos importantes sobre projeto conceitual de banco de dados, as ferramentas do modelo ER e DER e as regras ou restrições de integridade do negócio (RIs).

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Identificar relacionamentos dos tipos unário, binário e ternário;
2. Definir generalizações e especializações de entidades;
3. Descrever o conceito de entidade associativa;
4. Reconhecer as restrições de integridade (RIS) e seu funcionamento.

Vamos em frente...

A esta altura, você deve estar se perguntando o que mais precisa conhecer sobre modelo conceitual. Calma, falta pouco para você conhecer tudo de que precisa para ser um *expert* no modelo conceitual de banco de dados relacional. Nesta aula, vamos mostrar a você como interpretar situações incomuns que podem ocorrer no mundo real. Você também vai aprender como documentar regras específicas da realidade a ser modelada que não conseguimos traduzir para dentro do modelo ER.

Preparado para mais uma etapa? Vamos lá!

Fim da introdução

Tipos de relacionamentos

Até agora você viu diversos casos de relacionamento ocorrendo sempre entre duas entidades. Essa situação, em que duas entidades diferentes se relacionam, é a mais comum no modelo conceitual. Contudo, existem casos

menos comuns em que pode haver relacionamentos envolvendo apenas uma única entidade ou ainda mais de duas entidades. Você verá esses casos especiais a seguir.

Relacionamento unário (ou autorrelacionamento)

É o relacionamento entre instâncias da mesma entidade. Na Figura 4.1 temos o relacionamento Supervisiona, que ocorre entre dois empregados, em que um deles é o supervisor e o outro é o supervisionado. Repare que ambos pertencem à entidade Empregado.

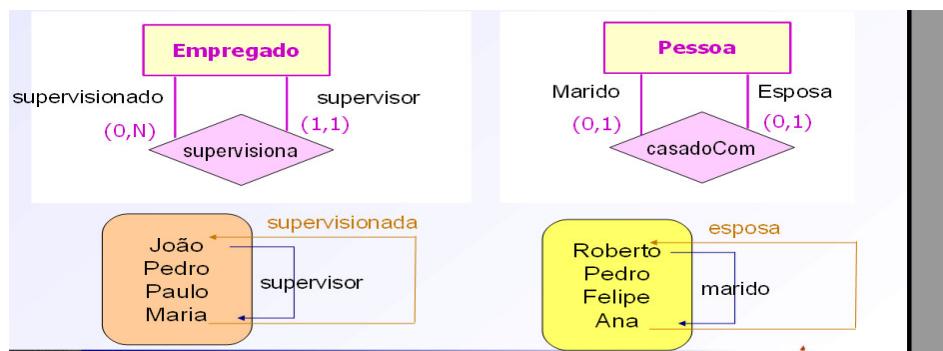


Figura 4.1. Relacionamento unário

Relacionamento binário

É aquele que envolve duas instâncias de entidades.

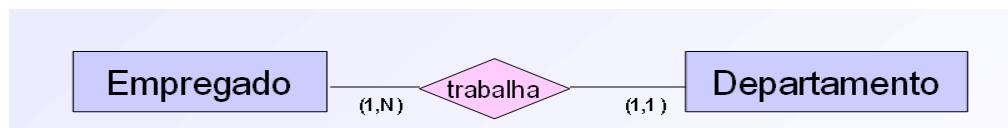


Figura 4.2. Relacionamento binário

- ✓ Um empregado pode trabalhar, no mínimo, em um departamento e, no máximo, em N departamentos;
- ✓ Um departamento pode trabalhar, no mínimo, com um empregado.

Ainda não ficou claro? Vamos ver mais alguns exemplos.

Exemplo 1: Observe a classe de relacionamentos no seguinte diagrama de

entidades:

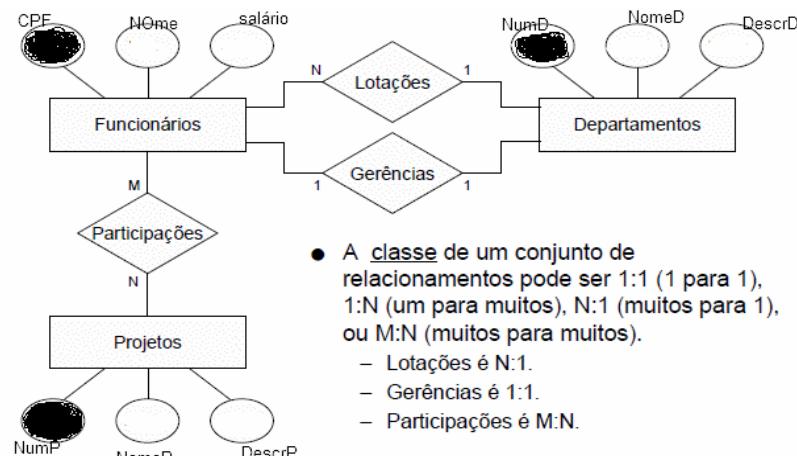


Figura 4.3. Classe de relacionamentos

- ✓ Um funcionário pode participar em N projetos e cada projeto pode ter N funcionários;
- ✓ Um funcionário está lotado em um determinado departamento e um departamento possui N funcionários;
- ✓ Um funcionário gerencia um departamento e um departamento é gerenciado por um funcionário.

Relacionamento ternário

Representa o relacionamento entre três entidades. Neste caso, todas as entidades ocorrem simultaneamente, ou seja, todas as instâncias do relacionamento possuem ligações com todas as entidades envolvidas no relacionamento.

Para analisar a cardinalidade em um relacionamento ternário, devemos levar em conta sempre a correspondência de duas entidades existentes comparando-as com uma terceira entidade, de forma a avaliar as cardinalidades mínima e máxima do relacionamento dessa entidade para com a dupla de entidades em questão.

Exemplo:

Um aluno inscrito em uma determinada disciplina (instância da dupla de

entidades aluno e disciplina) só terá um único professor. Ou seja, dando nome aos bois, se considerarmos o aluno Marcos tendo aula da disciplina de Banco de Dados, para esta situação só existiria um único professor: André, por exemplo. Isso justifica o número 1 que aparece perto da entidade professor. Já um professor que leciona uma disciplina pode estar lecionando para um ou mais alunos. Isso pode ser indicado pela letra N que aparece perto da entidade aluno. Já um aluno que recebe aulas de um professor pode estar recebendo essas aulas de uma ou mais disciplinas. Isso justifica a letra N que aparece perto da entidade disciplina.

Cada par de instâncias (aluno e disciplina) está associado, no máximo, a uma disciplina ou um professor pode ministrar várias disciplinas a um determinado aluno.

Um par (disciplina e professor) pode estar associado a muitos alunos ou um professor pode ministrar uma determinada disciplina a vários alunos.

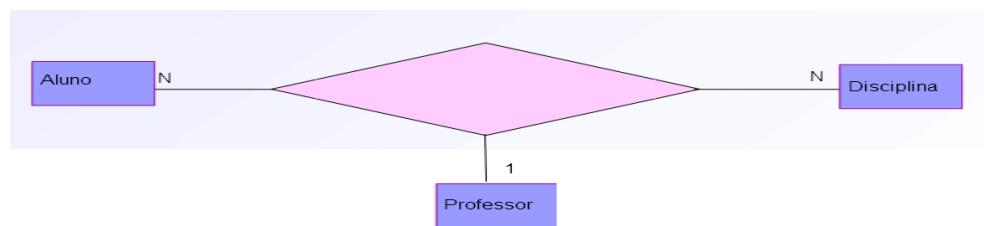


Figura 4.4. Relacionamento Ternário

Exemplo 2: Vamos interpretar o seguinte diagrama de entidades:

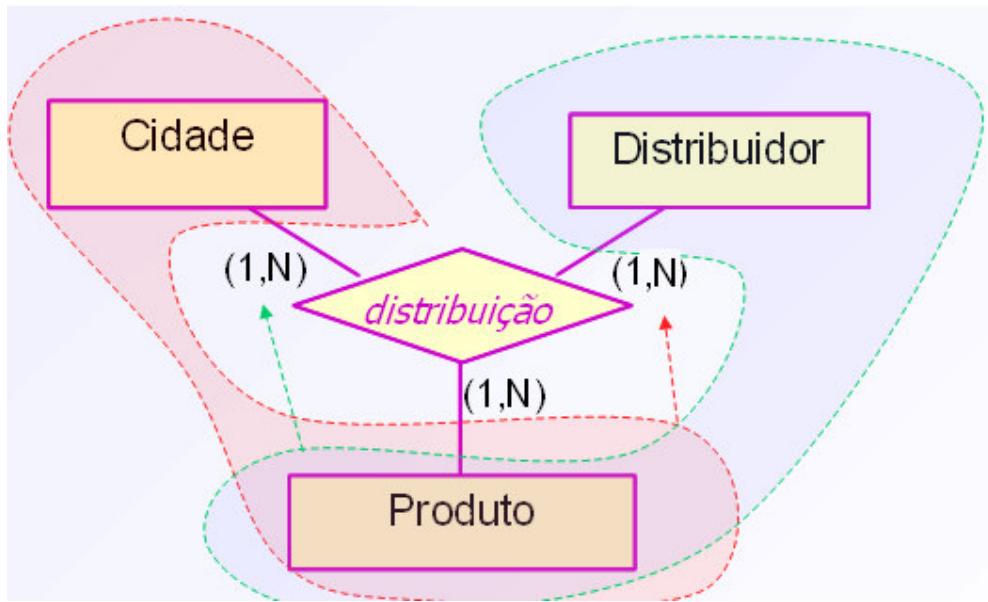


Figura 4.5. Exemplo de Classe de Relacionamento Ternário

- ✓ A distribuição de um produto em uma determinada cidade é feita por um ou mais distribuidores ($1 \times N$);
- ✓ Um produto, sendo distribuído por um único distribuidor, pode ser oferecido em, no mínimo, uma cidade e, no máximo, em N cidades;
- ✓ Um distribuidor que atua na cidade pode estar distribuindo de 1 a N produtos,

Início da atividade

Atividade 1 - Atende ao objetivo 1

Analise o minimundo a seguir e liste as entidades envolvidas e seus respectivos atributos. Em seguida, construa um diagrama ER apresentando os relacionamentos existentes entre as entidades que você encontrou, identificando que tipo de relacionamento ocorre (unário, binário, ternário).

Minimundo 07: Jogos

1. O *XGamer* é um sistema desenvolvido para realizar o cadastro de jogadores de *videogames* (consoles) de 3^a geração (Playstation 2 e 3, xBox360 e Nintendo Wii). O sistema precisa manter informações sobre quais jogos cada jogador cadastrado jogou em cada console nos últimos anos, incluindo datas

de conclusão e avaliação de cada jogo.

2. Dados completos sobre jogos e consoles (incluindo identificador, nome do console, fabricante e localização do fabricante) também devem ser acessados pelo sistema, fornecendo, assim, informações mais detalhadas aos jogadores.

As principais funcionalidades esperadas desse sistema são:

- a. cadastro de jogadores (identificador, nome, apelido, idade);
- b. cadastro dos jogos concluídos por um jogador (ano em que jogou e nota);
- c. consulta dos dados de um jogador;
- d. consulta de todos os jogos concluídos por um jogador (através do seu apelido);
- e. consulta de jogos por categoria, incluindo gênero e classificação;
- f. listagem de todos os jogos armazenados no sistema, incluindo: identificador e nome do jogo, ano de lançamento do jogo e código, nome e localização da produtora associada ao jogo;
- g. listagem de todos os jogadores cadastrados.

Resposta comentada

Após analisar o minimundo, efetuamos os seguintes passos:

Passo 1: Identificar as entidades com seus respectivos atributos

Entidade Jogador: repare que este “é um sistema desenvolvido para realizar o cadastro de jogadores”. A importância de manter informações sobre o jogador pode ser visualizada pela primeira funcionalidade esperada do sistema: - cadastro de jogadores (identificador, nome, apelido, idade);

Seus atributos são: código de identificação, nome do jogador, apelido e idade.

Entidade Jogo: repare o texto no parágrafo 2 “Dados completos sobre jogos também devem ser acessados através do sistema” e a funcionalidade solicitada na letra f. “listagem de todos os jogos armazenados no sistema, incluindo: identificador e nome do jogo, ano de lançamento do jogo e código, nome e localização da produtora associada ao jogo”, mostrando a importância das informações sobre jogos;

Seus atributos são: número de identificação do jogo e nome do jogo.

Entidade Console: repare que console possui várias informações relevantes para o sistema, como pode ser visto pelo texto da funcionalidade “consulta de jogos por categoria, incluindo gênero e classificação”.

Seus atributos são: número de identificação do console e nome do console.

Entidade Categoria: repare que categoria possui informações relevantes para o sistema, como pode ser visto no item e. “consulta de jogos por categoria, incluindo gênero e classificação”.

Seus atributos são: identificação da categoria, gênero e classificação.

Entidade Fabricante: repare que fabricante possui informações relevantes para o sistema, como pode ser visto no parágrafo 2: “Dados completos sobre jogos e consoles (incluindo identificador, nome do console, fabricante e localização do fabricante) também devem ser acessados através do sistema”.

Seus atributos são: identificação do fabricante, nome e localização.

Entidade Produtora: repare que produtora possui informações relevantes para o sistema, como pode ser visto no item f. “listagem de todos os jogos armazenados no sistema, incluindo: identificador e nome do jogo, ano de lançamento do jogo e código, nome e localização da produtora associada ao jogo”.

Seus atributos são: identificação da produtora, nome e localização.

Passo 2: Identificar quais são os relacionamentos entre as entidades e seus respectivos atributos

Relacionamento Joga: repare que há necessidade de o sistema armazenar “quais jogos cada jogador jogou em cada console”. Ao analisar essa frase, podemos perceber a presença de três entidades: Jogo, Jogador e Console. Esta relação envolve as três entidades ao mesmo tempo, dando origem então a um relacionamento do tipo ternário.

Relacionamento Enquadra-se: cada jogo se enquadra em determinada categoria. Isso pode ser evidenciado pelo texto “consulta de jogos por

categoria, incluindo gênero e classificação” do item e;

Relacionamento Distribuído: os jogos são distribuídos pelas produtoras, o que fica evidenciado pelo texto do item f “listagem de todos os jogos armazenados no sistema, incluindo: identificador e nome do jogo, ano de lançamento do jogo e o código, o nome e a localização da produtora associada ao jogo”. Este relacionamento possui o seguinte atributo: ano de lançamento do jogo.

Relacionamento Produzido: os consoles são produzidos por fabricantes. Isto pode ser observado através do texto “Dados completos sobre jogos e consoles (incluindo identificador, nome do console, fabricante e localização do fabricante) também devem ser acessados através do sistema, fornecendo, assim, informações mais detalhadas aos jogadores”.

Passo 3: Identificar as cardinalidades mínima e máxima de cada relacionamento

Relacionamento Joga: cada jogador pode jogar seu jogo em N consoles; daí a cardinalidade (1,N) em console. Cada jogador, com seu console, pode jogar de 1 a N jogos, daí a cardinalidade (1,N). Um jogo rodando num console pode ser jogado por 1 a N jogadores, daí a cardinalidade (1,N).

Box de atenção

Diferentemente do relacionamento binário, no ternário devemos olhar sempre uma dupla de instâncias de entidades em relação a uma terceira entidade. Isso deve ser feito para cada uma das entidades presentes no relacionamento.

Fim do Box de atenção

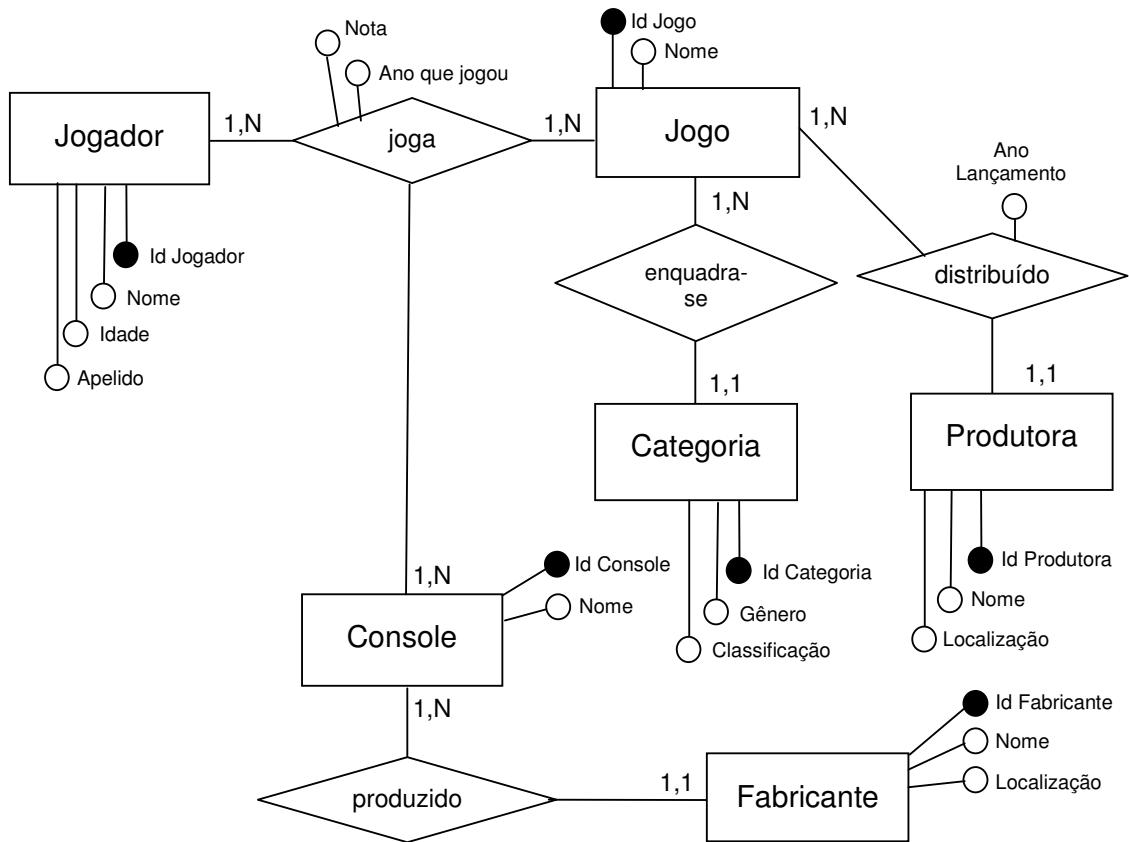
Relacionamento Enquadra-se: cada jogo se enquadra em apenas uma categoria (1,1) e cada categoria pode abranger de 1 a N jogos, daí a cardinalidade (1,N).

Relacionamento Distribuído: cada jogo é distribuído por uma única produtora, daí a cardinalidade (1,1); cada produtora pode distribuir de 1 a N jogos, daí a cardinalidade (1,N).

Relacionamento Produzido: um console é produzido por apenas um fabricante, daí a cardinalidade (1,1). Por outro lado, cada fabricante pode produzir de 1 a N consoles, gerando a cardinalidade (1,N).

Passo 4: Elaborar o DER

O DER representando o minimundo descrito deve ser elaborado considerando as entidades e seus respectivos atributos, o relacionamento entre elas com os atributos desses relacionamentos; finalmente, ele deve expressar a cardinalidade de cada relacionamento. O DER deve, então, refletir todas as informações relevantes que foram capturadas a partir da descrição do mundo real.



Fim da atividade

Especialização e generalização de entidades

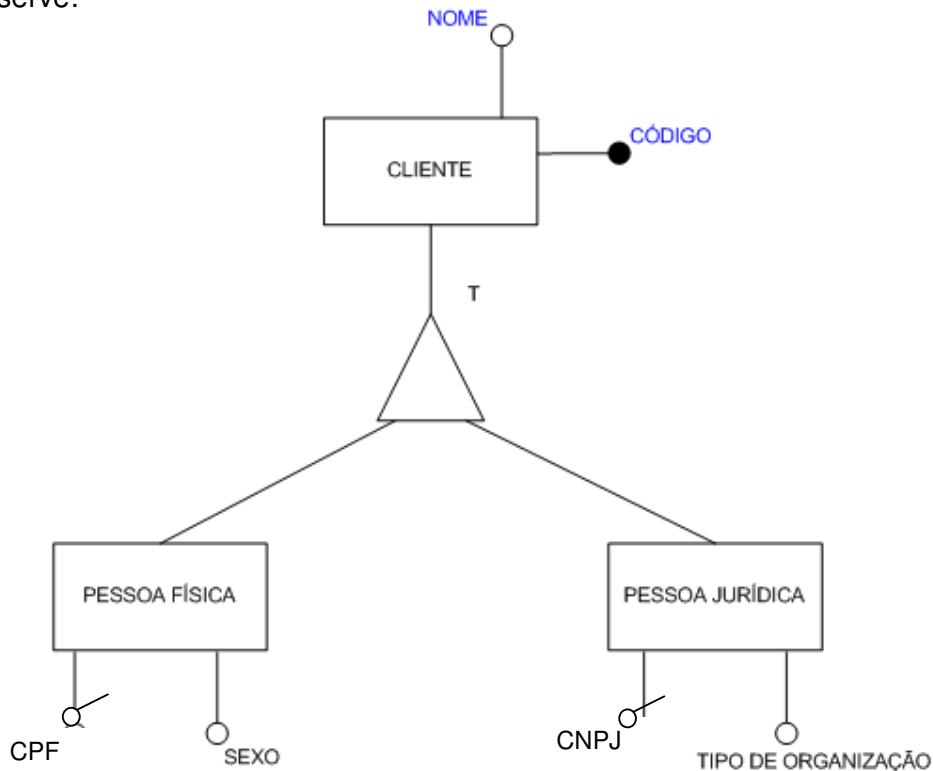
Imagine uma situação em que você é uma empresa que atende a clientes do

tipo pessoa física ou pessoa jurídica. Alguns dados dos clientes, como nome e código, são comuns tanto para pessoa física como jurídica. No caso de uma pessoa física, temos dados específicos, como o CPF e o sexo. Já no caso de pessoa jurídica, temos o CNPJ e o tipo de organização.

Então, como podemos expressar isso através de um modelo conceitual?

No mecanismo de generalização, atributos comuns a entidades de mais baixo nível hierárquico são representados uma única vez na entidade de mais alto nível. Note no exemplo a seguir que a entidade Cliente (genérica) é considerada de mais alto nível hierárquico que as entidades Pessoa Física e Pessoa Jurídica (especializadas), por ser a generalização das mesmas.

Observe:



Existem dois tipos de generalização/especialização: total (representada pela letra **T**) e parcial (representada pela letra **P**):

- na total, para cada ocorrência da entidade genérica existe sempre alguma ocorrência em uma das entidades especializadas.

Perceba que, no caso do exemplo dos clientes, ocorre uma generalização/especialização do tipo total, uma vez que todos os clientes ou são pessoas físicas ou jurídicas. Não existe um cliente que não esteja dentro de um desses dois tipos.

- na parcial, nem toda ocorrência da entidade genérica corresponde a uma entidade especializada.

Veja que, no exemplo do funcionário, nem todos os funcionários serão obrigatoriamente engenheiros ou médicos. Pode haver um funcionário sem as especializações de médico ou engenheiro. Nestes casos, a generalização/especialização é considerada parcial.

Entidade associativa

Observe agora outra situação: um paciente vai ao médico, que realiza uma consulta. Após a consulta, o médico prescreve alguns medicamentos que o paciente deve tomar. Como podemos expressar essa situação por meio de um modelo conceitual?

Considerando o médico, o paciente e o medicamento como entidades, perceba que, neste caso, o medicamento não estará direta nem exclusivamente associado ao médico ou ao paciente, mas sim ao resultado do encontro do médico com o paciente, que é a consulta. Teremos então a entidade medicamento associada ao relacionamento consulta.

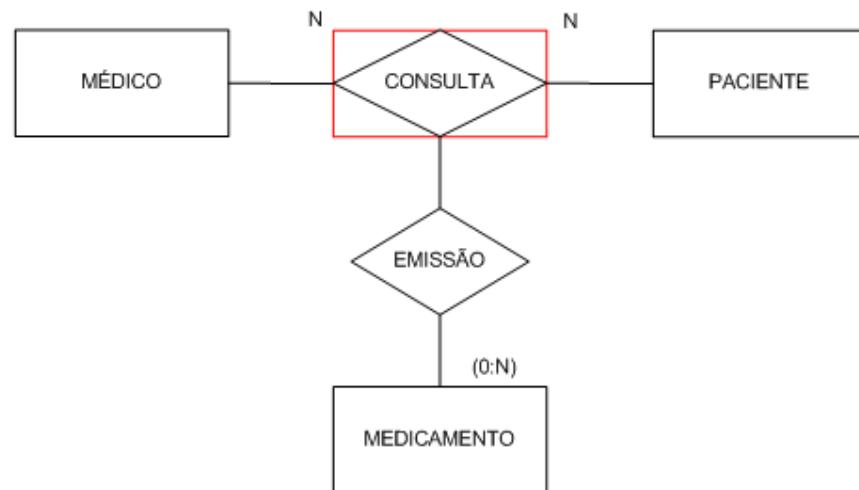
Em alguns casos, é necessário que associemos uma entidade à ocorrência de um relacionamento. O modelo de entidades e relacionamentos não permite relacionamentos entre relacionamentos, somente entre entidades. A ideia da entidade associativa é tratar um relacionamento como se ele fosse uma entidade. Observe o modelo a seguir:



Se desejarmos controlar os medicamentos receitados pelo médico em determinada consulta, temos que relacionar a entidade medicamento ao fato de ter havido uma consulta (relacionamento consulta).



Como não podemos fazer isso diretamente, indicamos que o relacionamento consulta é uma entidade associativa por meio de um retângulo em volta do relacionamento. Isso faz com que o relacionamento funcione como se fosse uma entidade, permitindo que haja relação com outro relacionamento.



Neste caso, para determinada consulta, pode haver ou não a emissão de receitas para medicamentos.

Atividade 2 - Atende aos objetivos 2 e 3

A partir da leitura e análise do minimundo a seguir, liste as entidades envolvidas e seus respectivos atributos. Em seguida, construa um diagrama ER apresentando os relacionamentos existentes entre as entidades que você encontrou, identificando que tipo de relacionamento ocorre (unário, binário, ternário), bem como as entidades associativas.

1. A Farmácia Curesaki deseja reduzir seus custos com a negociação e compra de medicamentos de fornecedores em grande escala. Além da venda de medicamentos, a farmácia também possui uma seção de artigos de perfumaria que se encontra localizada no interior da loja e é responsável por uma parcela significante do faturamento total com os produtos.
2. Alguns medicamentos precisam de uma receita médica para que possam ser comercializados no interior do estabelecimento, sob pena de multa ou fechamento da farmácia. Nesses casos, os dados da receita médica (médico, CRM, data, prescrição do medicamento) precisam estar cadastrados no sistema para liberar a venda.
3. O dono da rede de farmácias associadas deseja implantar um sistema integrado que atenda às necessidades descritas e seja capaz de produzir as seguintes listagens:
 - a. Relação de fornecedores de produtos, contendo: código, nome, endereços e telefones de cada fornecedor cadastrado;
 - b. Relatório das vendas mensais, discriminando, para cada venda listada, valor total, data, operador e número da nota fiscal;
 - c. Relação mensal dos produtos mais vendidos na loja, indicando o tipo de produto (medicamento ou artigo de perfumaria), código, nome do produto, quantidade vendida e valor total ganho com o produto naquele mês;
 - d. Relatório de produtos com estoque disponível, contendo: código, nome do produto e estoque. Para o caso de medicamentos, é necessário incluir também a validade e indicador de tarja preta. No caso de perfumes, incluir a fragrância e o conteúdo líquido (mL).

Fim da atividade online

Restrições de integridade

O poder de expressão dos modelos ER é limitado. Nem todas as propriedades de um banco de dados são apresentadas no modelo ER, haja vista que a linguagem do modelo é limitada. Portanto, restrições de integridade do minimundo devem ser capturadas.

Falando no mundo dos negócios, as restrições de integridade são aquelas

regras de negócio não capturadas pelo DER, mas que precisam ser especificadas a título de completar o trabalho. Por exemplo, o salário de um subordinado não pode ser maior do que o de seu chefe. Ou seja, restrição de integridade é uma regra que é estabelecida pela realidade modelada e que deve ser obedecida pelo banco de dados.

Outros exemplos são as restrições de integridade do minimundo sobre jogos descrito na Atividade 1. Temos as restrições de integridade que devem ser documentadas em linguagem natural, porque não conseguem ser expressadas completamente dentro do modelo ER. Essas restrições são:

- um jogador não poderá cadastrar um jogo que ainda não foi lançado;
- um jogo possui classificação etária; logo, para cadastrar um jogo, a idade do jogador deverá ser compatível com a sua classificação;
- as classificações devem estar entre 5 e 18 anos;
- as notas atribuídas a um jogo devem estar compreendidas entre 0 e 10.

Agora vamos praticar! A próxima atividade requer a aplicação de todos os conceitos vistos nesta aula.

Início de Atividade online

Atividade 3 - Atende aos objetivos 1, 2, 3 e 4

Elabore o DER completo que reflita o minimundo apresentado e relacione também as restrições de integridade que devem ser consideradas para a implementação do sistema a ser desenvolvido.

Minimundo 09: Sistema de vendas e faturamento da Oxxen

Você trabalha na empresa de distribuição de petróleo Oxxen. Seu chefe imediato o alocou para fazer um levantamento dos requisitos de informação (necessidades) para a especificação de um novo sistema de controle de vendas e faturamento para a empresa.

Após entrevistar os funcionários, você obteve as seguintes informações:

1. A empresa Oxxen possui dois tipos de cliente: varejo, que inclui os postos de serviço; e industrial, que comercializa produtos derivados do petróleo, como óleos lubrificantes. Os postos de serviço possuem capacidade total de

armazenamento dos tanques e o número total de frentistas que atendem no posto. Os clientes industriais são responsáveis pela venda dos produtos embalados. Estes possuem um tamanho físico do depósito onde estocam os produtos e possuem um número total de vendedores que pode variar de acordo com o crescimento das vendas. Para ambos os casos, devem ser armazenados também o código, o nome, os endereços e telefones do cliente.

2. Os postos de serviço, quando necessitam repor seus estoques, telefonam para a central de atendimento ao cliente e solicitam os produtos e respectivas quantidades a serem entregues.

3. Logo que o atendente verifica que se trata de um cliente cadastrado, ele aloca um transportador que virá buscar o produto nos armazéns da empresa e irá entregá-lo no local especificado pelo cliente. Após confirmar a viabilidade do transportador e o valor do frete para a entrega, o atendente entra no sistema e cadastrá uma venda. Ao cadastrá-la, é obrigatório registrar no sistema o atendente e o cliente associados. Em seguida, o atendente emite uma nota fiscal que contém a data da venda, o código e nome do cliente, seu endereço para entrega, produtos, quantidades solicitadas e valor total da venda.

4. Cada nota fiscal/fatura emitida conterá todos os produtos (com respectivos códigos e descrição) que foram solicitados para a entrega, bem como o código do transportador que realizará a entrega. O frete referente ao percurso também consta no corpo da nota-fiscal/fatura. As entregas só podem ser realizadas por transportadores autorizados pela empresa e que constem em seu cadastro. A quantidade do produto a ser entregue não pode exceder a capacidade do veículo transportador.

5. O sistema precisa monitorar as vendas de cada atendente, pois a sua remuneração ao final do mês pode variar em função delas. Nenhum atendente poderá receber mais do que três vezes o seu próprio salário em comissões no mês.

6. Os usuários desejam que o sistema solicitado possa lhes informar, diária e mensalmente, todas as vendas efetuadas listando os clientes, os valores de cada nota fiscal e o total do faturamento.

O sistema também precisará gerar as seguintes listagens/relatórios:

a. Listagem de produtos com estoque disponível na empresa, incluindo código

do produto, nome e peso.

- b. Listagem das transportadoras que prestam serviço à empresa, relacionando código, nome, localidade e telefones.
- c. Relatório mensal de atendentes mais eficientes, incluindo código do atendente, nome, salário, telefones, porcentagem de comissão combinada por venda e valor total recebido com comissões de vendas realizadas no mês.

Fim da Atividade online

Conclusão

Com esta aula, finalizamos a primeira parte do projeto de banco de dados: a modelagem conceitual. Esta parte é a mais importante para a construção de um bom banco de dados, já que todas as demais etapas serão desenvolvidas com base no modelo conceitual.

Resumo

Os relacionamentos podem ser:

- a) Unário (ou Autorrelacionamento): relacionamento entre instâncias da mesma Entidade;
- b) Binário: é aquele que envolve duas instâncias de Entidade;
- c) Ternário: relacionado entre múltiplas Entidades. Expressam um fato em que todas as entidades ocorrem simultaneamente, ou seja, todas as instâncias do relacionamento sempre possuem ligações com todas as entidades envolvidas no relacionamento.

No mecanismo de generalização, atributos comuns a entidades de mais baixo nível são representados uma única vez na entidade de mais alto nível.

Existem dois tipos de generalização/especialização: total e parcial.

- na total, para cada ocorrência da entidade genérica existe sempre ocorrência em uma das entidades especializadas.
- na parcial, nem toda ocorrência da entidade genérica corresponde a uma entidade especializada.

Entidade associativa: em alguns casos, é necessário que associemos uma entidade com a ocorrência de um relacionamento. O modelo de entidades e

relacionamentos não permite relacionamentos entre relacionamentos, somente entre entidades. A ideia da entidade associativa é tratar um relacionamento como se ele fosse uma entidade.

O poder de expressão dos modelos ER é limitado. Nem todas as propriedades de um banco de dados são apresentadas no modelo ER, haja vista que a linguagem do modelo é limitada. Falando no mundo dos negócios, as restrições de integridade são aquelas regras de negócio não capturadas pelo DER, mas que precisam ser especificadas a título de completeza. Por exemplo, o salário de um subordinado não pode ser maior do que o de seu chefe.

Informações sobre a próxima aula

A transformação do projeto conceitual abstrato para o projeto lógico, formalizando os conceitos do Modelo ER, será o tema da próxima aula.

Então faremos a modelagem relacional como passo seguinte para a construção ou criação de um banco de dados relacional.

Referências bibliográficas

CHEN, P. *Modelagem de dados. A abordagem entidade-relacionamento para projeto lógico.* São Paulo: McGraw-Hill e Makron Books do Brasil, 1990.

ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados.* 5^a ed. São Paulo: Pearson Addison Wesley, 2009.

Curso de Extensão: Modelando e Implementando Bancos de Dados Relacionais
Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 5

Projeto lógico - modelo relacional

Meta

Apresentar os conceitos importantes e as principais características de um projeto lógico, assim como o mapeamento do modelo conceitual para utilização do modelo relacional em banco de dados.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

Descrever os conceitos básicos do modelo relacional;

Estabelecer os seguintes mapeamentos do modelo conceitual para o modelo relacional:

- Mapeamento de relacionamento um para um (1:1);
- Mapeamento de relacionamento um para muitos (1:N);
- Mapeamento de relacionamento muitos para muitos (N:N);
- Mapeamento de relacionamento binário (cardinalidade máxima-mínima);
- Mapeamento de atributos multivvalorados.

Pré-requisitos

Para o correto e completo aprendizado dessa aula, é importante você relembrar:

- os conceitos de banco de dados vistos na Aula 1;
- os conceitos de projeto de banco de dados, projeto conceitual, levantamento de requisitos e regras de negócio, estudados na Aula 2;
- os modelos ER e DER e os conceitos de cardinalidade mínima e atributo multivvalorado, apresentados na Aula 3.

Os bancos de dados relacionais vencem

Você já viu que o projeto lógico constitui a fase posterior à modelagem conceitual do banco de dados. Ele procura detalhar a forma como as entidades, atributos e relacionamentos serão apresentados no banco de dados relacional. Por esse motivo, depende da tecnologia que será utilizada. Em um ambiente de banco de dados relacional utilizamos alguns conceitos importantes. Diversos desses termos originaram-se diretamente da teoria de conjuntos; outros, das operações desenvolvidas especificamente para os bancos de dados relacionais.

Por outro lado, dentro da categoria de modelos lógicos, existem vários tipos de modelos de dados; o mais utilizado atualmente é o **modelo relacional**. Nesta aula você verá com detalhes as principais características desse modelo de banco de dados.

Fim da introdução

Início do Box de Curiosidade

Modelo relacional

O conceito de modelo relacional foi criado por Edgar Codd em 1970; foi descrito no artigo *Relational model of data for large shared data banks*. Historicamente, esse modelo é o sucessor do modelo hierárquico e do modelo de rede. Essas arquiteturas antigas são utilizadas até hoje em alguns *data centers* com alto volume de dados, em que a migração é inviabilizada pelo custo que demandaria. Existem ainda os novos modelos baseados em orientação ao objeto: trata-se de novo paradigma na utilização do banco de dados.

Fim do box

Conceitos de banco de dados relacional

Veja a seguir alguns conceitos básicos relacionados a banco de dados relacional:

- **Modelo de dados relacional**

Nesse modelo, todos os dados estão guardados em tabelas ou relações. Toda a sua

definição é teórica e baseada em lógica, **álgebra relacional** e teoria de conjuntos. A álgebra relacional é muito importante, pois seus conceitos são incorporados na linguagem de consulta-padrão SQL.

Início de Verbete

Álgebra relacional

Os operadores da álgebra relacional recebem uma ou duas relações ou tabelas como operandos e produzem uma nova relação como resultado.

Fim do Verbete

As seguintes definições são importantes para a compreensão do modelo de dados relacional:

1. Relação ou tabela:

- É a estrutura que armazena os dados reais referentes às entidades e/ou relacionamentos definidos no modelo conceitual. Então, ao projetarmos um banco de dados relacional a partir de um modelo ER, criamos uma tabela para cada entidade existente ou relacionamento que possua atributos;
- As tabelas são também chamadas de relações (na teoria de conjuntos);
- Uma tabela é composta por colunas (campos) e linhas (registros);
- Toda tabela possui um nome que normalmente corresponde ao nome da entidade da qual ela se originou. Esse nome aparece sempre no singular. Exemplos de nomes para tabelas: Cliente, Produto, Venda.

2. Coluna ou atributo:

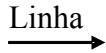
- Cada tipo de informação armazenada numa tabela é uma coluna. Toda coluna de uma tabela deve possuir um nome pelo qual será referenciada. Exemplos de nomes para colunas ou campos que poderiam compor a tabela Cliente: Nome, CPF, Endereço, Telefone, E-mail;
- Todos os valores numa coluna pertencem ao mesmo tipo de dado;
- Assim como você viu que as entidades correspondem às tabelas, os atributos de uma entidade dão origem aos campos ou colunas da tabela em questão;

- Cada coluna representa um campo existente na tabela, ou seja, uma área onde serão armazenados dados de um determinado tipo.

3. Linhas ou tuplas:

- Cada linha representa uma coleção de dados relacionados a um registro existente na tabela. Exemplo de linha ou registro para a tabela Cliente:

Tabela Cliente



| Nome | CPF | Endereço | Telefone | E-mail |
|----------------|--------------|-------------------------|----------------|---------------------|
| Pedro Assunção | 073682345-18 | Rua das Ostras, 23 - RJ | (21) 4529-2382 | pedroass@bol.com.br |

4. Domínio:

- Conjunto de valores possíveis (válidos) em cada coluna.

O modelo relacional representa um banco de dados como um conjunto de relações, tuplas (linhas), atributos (colunas) e um conjunto de valores possíveis em cada coluna (domínios).

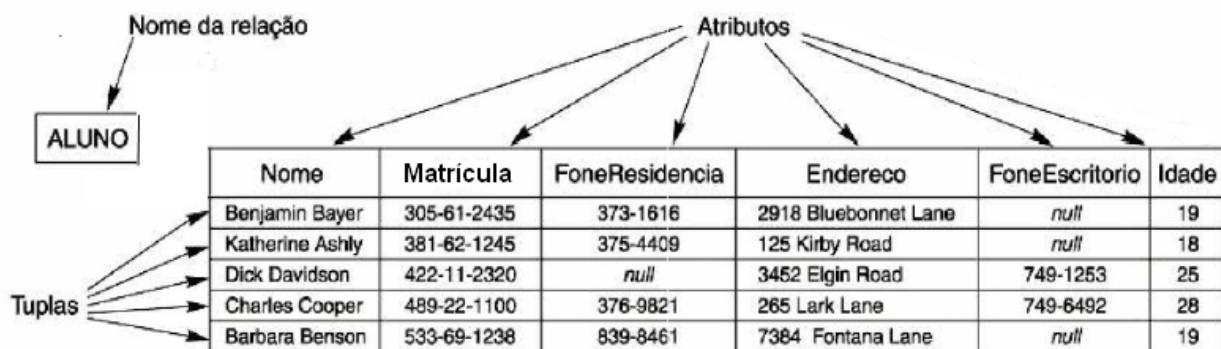


Figura 5.1: Exemplo: conceitos importantes do modelo relacional.

- **Tipos de chaves**

1. Chave primária:

- Um dos princípios do modelo relacional diz que uma linha de uma tabela deve

sempre pode ser referenciada de forma única;

- O atributo identificador que não se repete entre as linhas de uma tabela é chamado *chave primária*. No caso da tabela Aluno, por exemplo, poderia ser a coluna Matrícula, já que essa informação é única entre os alunos.

2. Chaves compostas:

- Muitas vezes, 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*.

3. Chave candidata ou alternativa:

- É aquela que satisfaz todos os requisitos de uma chave primária, mas não foi eleita como tal;
- A chave primária é escolhida dentre as chaves candidatas pelo projetista do banco de dados. A escolha de uma chave primária é obrigatória.

Vamos fazer um exemplo. Na Figura 5.1, podemos obter as seguintes informações:

Atributos: Nome, Matricula, FoneResidencia, Endereco, FoneEscritorio, Idade.

Chaves candidatas: Nome, Matrícula, Idade, Endereço, FoneResidencia, FoneEscritório, pois elas não se repetem na tabela.

Qual delas será escolhida como chave primária?

Nome? ==> não é uma boa ideia, pois pode acontecer de duas pessoas terem o mesmo nome.

FoneResidencia ==> não é boa ideia, pois possui atributo desconhecido, também pode mudar com o tempo e mais de um aluno pode possuir o mesmo número.

Endereço? ==> também não é uma boa escolha; e se dois alunos morarem juntos?

FoneEscritório ==> também possui atributo desconhecido e mais de um aluno pode possuir o mesmo número de telefone.

Matrícula? ==> é a melhor opção, pois há um único número para cada pessoa.
 Então, a chave primária é Matricula.

4. Chave estrangeira:

- É uma coluna (ou combinação de colunas) que indica um valor que deve existir como chave primária numa outra tabela. É a chave estrangeira que possibilita ligarmos os dados de uma tabela com outra. No caso da Figura 5.2, existem duas chaves estrangeiras: as colunas CodCargo e CodDept, que referenciam, respectivamente, as tabelas de cargos e de departamentos.

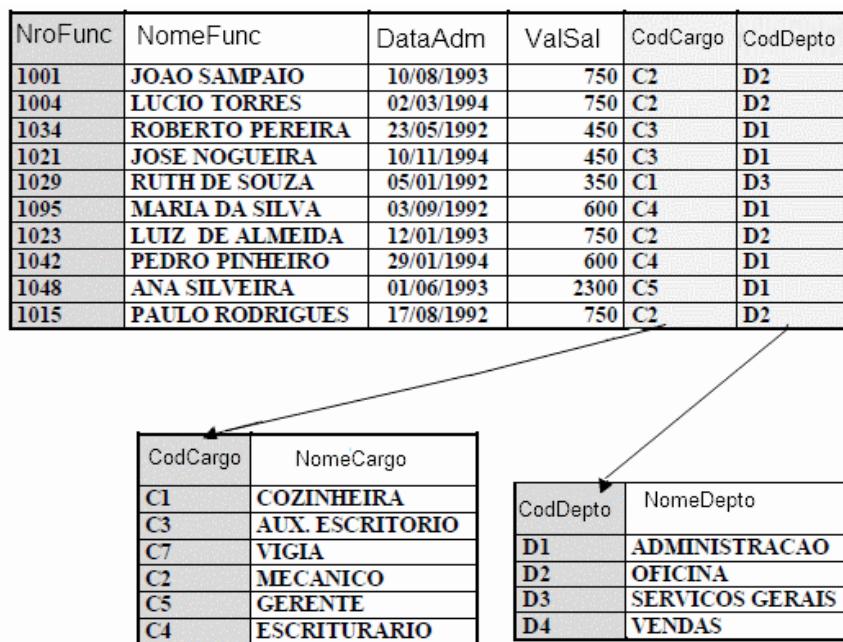


Figura 5.2: Exemplo de chaves estrangeiras para a tabela de funcionários.

- Conceito de nulo**
 - Representa um valor desconhecido para um dado. Não devemos confundir o conceito de nulo com espaços em branco ou o número zero;
 - 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 Aluno contiver um nulo na coluna Reservista, significa que código de reservista do aluno correspondente àquela linha é desconhecido.

The diagram shows a table with six columns: NrMatriC, NmAluno, (...), Sexo, RG, and Reservista. There are four rows of data. The first three rows have standard entries: Row 1 (NrMatriC 22941010, NmAluno CLAUDIO SILVA, Sexo M, RG 21212121, Reservista 91929); Row 2 (NrMatriC 22941012, NmAluno ANA RIBEIRO, Sexo F, RG 14141414, Reservista ?! ?! ?! ?!); and Row 3 (NrMatriC (...) (empty), NmAluno (...) (empty), Sexo (...), RG (...), Reservista (...)). A fourth row (NrMatriC 22941088, NmAluno JOSE SANTOS, Sexo M, RG 34343434, Reservista 828122) is shown below. Two arrows point from the text 'Espaços em branco' to the empty cells in the third row and from the text 'NULLO' to the question marks in the fourth row.

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

Figura 5.3: Exemplo de atributo com valor nulo permitido (Reservista).

Início da atividade

Atividade 1 - Atende ao objetivo 1

Considere a seguinte tabela:

| Nome | CPF | Nascimento | Endereço | Sexo | Salário | Chefe | Dept |
|---------------------------------|--------------|------------|--------------------------|------|--------------|--------------|------|
| Maria Aparecida | 146303469-62 | 20/7/1965 | Antônio Carlos 100 | Fem | R\$ 3.000,00 | 941261409-79 | 1 |
| Júlia Pires Nassi | 170499021-58 | 21/11/1968 | Costa e Lemos 401 | Fem | R\$ 2.000,00 | 941261409-79 | 2 |
| Liane Maria Alcina da Cunha | 458267321-14 | 7/2/1977 | Lemes de Brito 1003/13 | Fem | R\$ 1.000,00 | 490633506-81 | 3 |
| Alcides Maranhão Boaventura | 490633506-81 | 14/5/1974 | Gago Coutinho 1302 | Masc | R\$ 3.000,00 | 941261409-79 | 3 |
| Antônio Maria Jaguarão da Senna | 590084341-96 | 29/9/1977 | Mariana Coutinho 100/104 | Masc | R\$ 1.000,00 | 170499021-58 | 2 |
| Alves Nunes Coimbra | 688375441-79 | 16/12/1985 | Marta Rocha 322 | Masc | R\$ 2.000,00 | 490633506-81 | 3 |
| José da Silva | 875988400-39 | 20/7/1965 | Lopes Silva 307/401 | Masc | R\$ 3.000,00 | 941261409-79 | 1 |
| Márcio Fonseca | 941261409-79 | 3/5/1957 | Luís Costa 10/32 | Masc | R\$ 5.000,00 | nulo | 1 |

- a) Identifique os atributos.
- b) Identifique as chaves candidatas e analise a possível chave primária.

Resposta comentada

Observa-se o seguinte:

- α) Atributos: Nome, CPF, Nascimento, Endereço, Sexo, Salário, Chefe e Dept.
- β) Chaves candidatas: Nome, CPF, Endereço. Elas não se repetem na tabela.

Qual delas será escolhida como chave primária?

Nome? ==> não é uma boa ideia, pois pode acontecer de duas pessoas terem o mesmo nome.

Endereço? ==> também não é uma boa ideia, pois dois funcionários podem morar juntos.

CPF? ==> ótimo, já que há um único número de CPF para cada pessoa.

Então, a chave primária é o CPF.

Fim da atividade

Inicio da atividade

Atividade 2 - Atende ao objetivo 1

Identifique a chave estrangeira nas tabelas a seguir:

| NomeDep | NúmeroD | Gerente | DataInícioGerente |
|---------------|---------|--------------|-------------------|
| Administração | 1 | 941261409-79 | 1/1/2000 |
| Pesquisa | 2 | 170499021-58 | 2/4/2005 |
| Vendas | 3 | 490633506-81 | 1/10/2003 |

| NúmeroD | LocalID |
|---------|----------------|
| 1 | Rio de Janeiro |
| 2 | Rio de Janeiro |
| 3 | Rio de Janeiro |
| 3 | São Paulo |

Resposta comentada

Observamos duas tabelas: na primeira, a chave primária é o atributo NúmeroD; na segunda, a chave primária é o atributo também chamado NúmeroD. Quando realizamos o relacionamento entre essas chaves para a extração de informação, observamos que, num primeiro momento, o relacionamento de uma tabela com outra é feito através de suas chaves primárias; em seguida, procede-se à busca da chave primária NúmeroD da primeira tabela e se converte o atributo NúmeroD – chave estrangeira - da segunda tabela.

| NomeDep | NúmeroD | Gerente | DataInícioGerente |
|---------------|---------|--------------|-------------------|
| Administração | 1 | 941261409-79 | 1/1/2000 |
| Pesquisa | 2 | 170499021-58 | 2/4/2005 |
| Vendas | 3 | 490633506-81 | 1/10/2003 |

| NúmeroD | LocalID |
|---------|----------------|
| 1 | Rio de Janeiro |
| 2 | Rio de Janeiro |
| 3 | Rio de Janeiro |
| 3 | São Paulo |

Regras básicas de mapeamento de um modelo ER em um modelo relacional

No processo de criação do banco de dados, as etapas de modelagem passam de um nível conceitual para um nível de implementação basicamente passando do modelo ER para o modelo relacional. Para tal tarefa, existem algumas regras e os mais variados casos que podem ser seguidos, embora não cubram exatamente todas as

possibilidades. Agora, vamos apresentar as regras para obter o modelo relacional, que pode ser montado a partir das seguintes transformações do modelo conceitual:

1. Transformação das entidades e seus atributos em tabelas com seus respectivos campos;
2. Transformação dos relacionamentos e seus atributos em tabelas com seus respectivos campos.

1. Entidades e seus atributos

Qualquer componente do modelo conceitual que possua dados deve ser representado no banco de dados por meio de uma tabela. Isso significa que, a princípio, cada entidade ou relacionamento com atributos próprios deve corresponder a uma tabela do banco de dados.

- Cada atributo da entidade define uma coluna dessa tabela;
- Os atributos identificadores da entidade correspondem às colunas que compõem a chave primária da tabela. Exemplo:

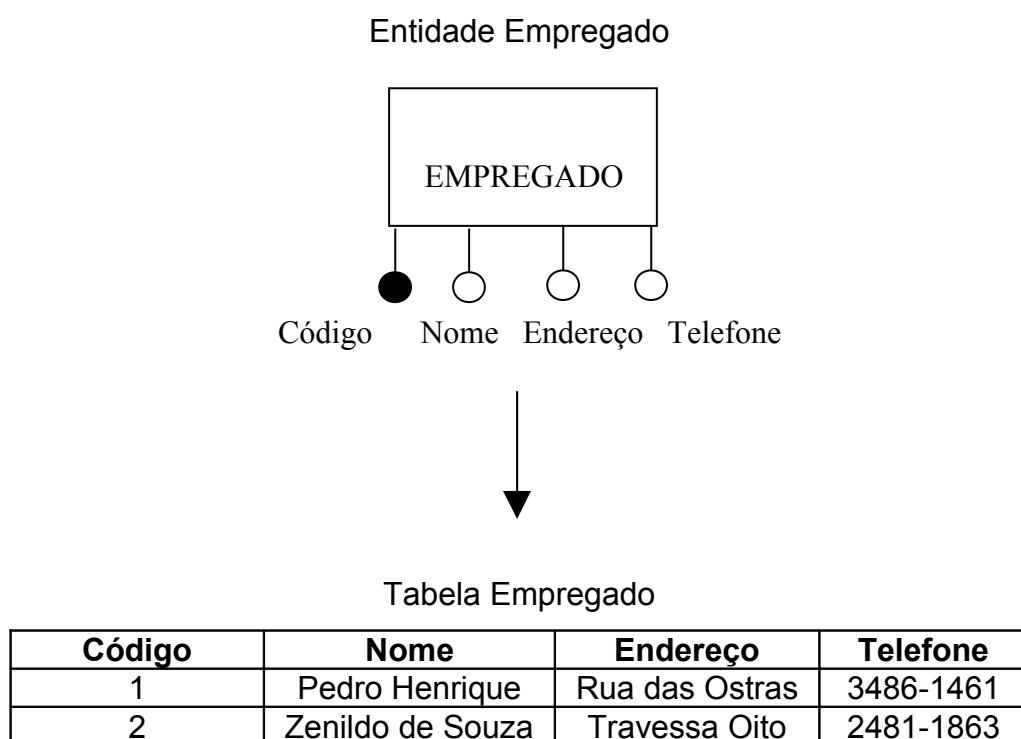


Figura 5.4: Entidade x Tabela.

A Figura 5.4 apresenta um exemplo da transformação de uma entidade em uma tabela. O diagrama entidade-relacionamento (DER) é o correspondente do esquema relacional. A entidade Empregado, com seus atributos: Código, Nome, Endereço e Telefone, é transformada na tabela denominada Empregado, com colunas denominadas Código, Nome, Endereço e Telefone. Como o atributo Código é o identificador da entidade, a coluna correspondente a esse atributo será a chave primária da tabela.

Nome de atributos e nomes de colunas

- As tabelas e colunas que constituirão o banco de dados devem receber nomes próprios;
- Um clássico exemplo de variação na nomenclatura entre o modelo conceitual e o modelo lógico é o do nomes das tabelas, que em geral são substantivos no plural, ao contrário do que ocorre com as entidades, costumeiramente batizadas com substantivos no singular;
- É conveniente manter os nomes das colunas curtos e claros. Assim, nomes de atributos compostos de diversas palavras devem ser abreviados;
- A nomeação de colunas é relativa quanto ao uso de abreviaturas. Muitas vezes, usam-se determinadas abreviaturas para tipos de campos que se repetem, como ID ou Id para identificação, Cod ou Cd para código, Nm para nome, End para endereço ou No ou Nr para número;
- Todo o esquema lógico deve ser documentado num **dicionário de dados**, indicando pelo menos o nome de cada objeto do banco de dados, sua finalidade e estrutura.

Início do Verbete

Dicionário de dados

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 se possui alguma regra de validação que deve ser considerada para o dado).

Fim do verbete

2. Relacionamentos e atributos

Como você já viu, 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.

- Algumas entidades podem ser fundidas em uma única tabela; outras podem ser desmembradas em duas ou mais tabelas distintas; o mesmo ocorre com o atributo multivalorado;
- Para cada atributo multivalorado: (i) criar uma nova relação; (ii) incluir o atributo correspondente ao atributo multivalorado mais a chave primária da relação que tem esse atributo; (iii) incluir como chave primária da nova relação a combinação do atributo multivalorado mais a chave primária da entidade que tinha o atributo multivalorado;
- Os relacionamentos também merecem alguma atenção quanto às várias possibilidades de implementação.

A seguir, vamos estudar os principais aspectos dos relacionamentos que devem ser considerados ao se projetarem as tabelas de um banco de dados relacional.

Relacionamentos 1 para 1 (1:1)

Sempre que um relacionamento de 1 para 1 é detectado, 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. Exemplo:

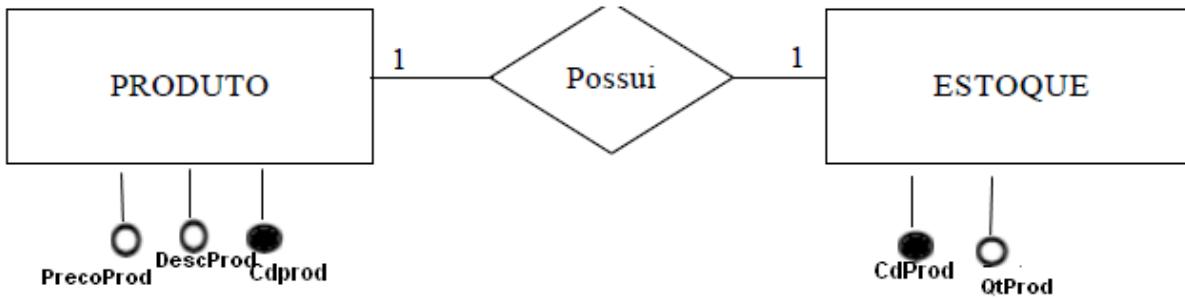


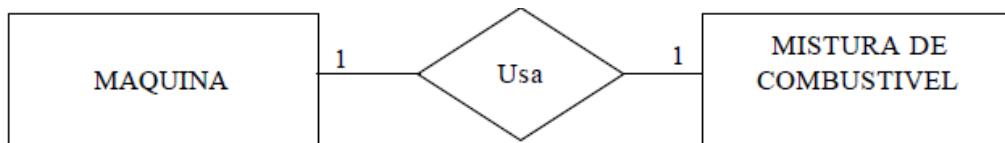
Figura 5.5: Relacionamento 1 para 1.

No diagrama da Figura 5.5 você pode observar que as entidades Produto e Estoque possuem o mesmo identificador. Logo, podemos representar as informações todas em uma só tabela (Produtos, no caso). Assim, temos o seguinte esquema relacional:

PRODUTO (CdProd, DescProd, PrecoProd, QtProd)

Repare que o campo CdProd está sublinhado. Isso significa que esse campo foi selecionado como chave primária da tabela. Essa notação será utilizada nos exemplos seguintes.

Se tivermos, como no diagrama a seguir, uma entidade Máquina e outra Mistura de Combustível, em que cada máquina tem sua própria mistura de combustível e cada mistura relaciona-se somente com uma máquina, devemos considerar que a mistura de combustível utilizada pode ser representada como uma coluna da tabela Máquinas.

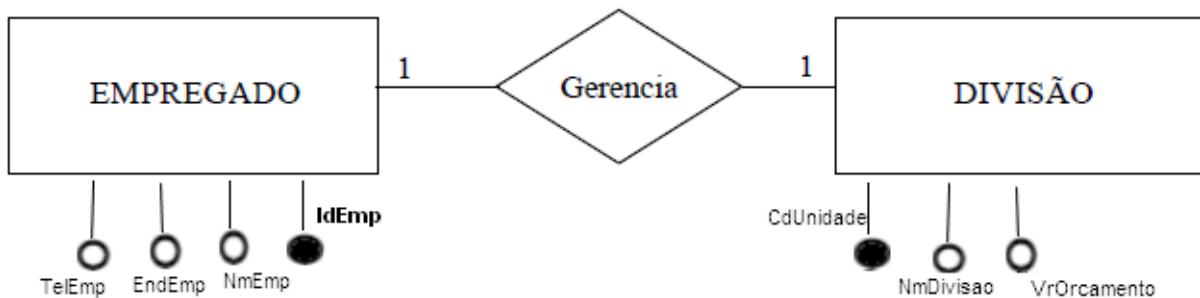


Dicas práticas:

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

Exemplo:

Suponha que temos um modelo retratando um relacionamento entre um funcionário e a divisão a que pertence, conforme o diagrama a seguir.



Expressando a entidade Empregado no seu esquema relacional:

EMPREGADO (IdEmp, NmEmp, EndEmp, TelEmp)

Expressando a entidade Divisão no seu esquema relacional:

DIVISÃO (CdUnidade, NmDivisao, VrOrcamento)

Para construir o relacionamento entre Empregado e Divisão, é necessário acrescentar uma coluna em uma das tabelas, que será utilizada como uma chave estrangeira, de forma a referenciar a outra tabela.

As soluções possíveis de distribuição da coluna da chave estrangeira são as seguintes:

EMPREGADO (IdEmp, NmEmp, EndEmp, TelEmp)

DIVISÃO (CdUnidade, NmDivisao, VrOrcamento, IdEmpGerente)

ou

EMPREGADO (IdEmp, NmEmp, EndEmp, TelEmp, CdUnidade)

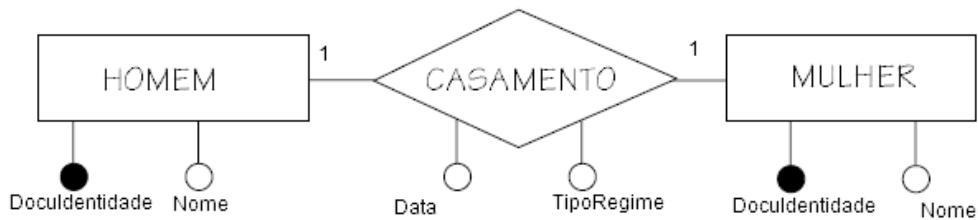
DIVISÃO (CdUnidade, NmDivisao, VrOrcamento)

A estrutura de colunas proposta à tabela Divisões, no primeiro caso, permite que no futuro um Empregado possa vir a gerenciar mais de uma divisão.

Início da atividade

Atividade 3 - Atende ao objetivo 2

Considere o seguinte DER e realize o mapeamento para o modelo relacional:



Resposta comentada

Neste diagrama "DER", o mapeamento pode ser feito da seguinte forma: temos duas entidades: a entidade Homem, com seus atributos Doculdade e Nome, e uma entidade Mulher, com seus atributos Doculdade e Nome. A questão é saber se as duas entidades são realmente distintas ou se elas podem ser unidas em uma única tabela. Nossa caso trata de entidades diferentes e com diferente atributo identificador. Por isso há uma forte razão para representar essas entidades em diferentes tabelas. O relacionamento se dará através de uma chave estrangeira.

Uma possível solução seria colocar na tabela Mulher um campo de chave estrangeira chamado *IdentH*, que seria o identificador do homem. Ainda na tabela Mulher, poderia ser feita a adição dos atributos de casamento Homem (DoculdH, NomeH) e Mulher (DoculdM, NomeM, Data, Regime, IdentH).

Dessa forma, poderiam ter sido adicionadas colunas de identificador da mulher e atributos de casamento à tabela Homem.

Fim da atividade

Relacionamentos 1 para muitos (1:N)

Para cada relacionamento 1:N no Esquema ER é preciso:

- (i) identificar a relação que representa a entidade do lado N;
- (ii) incluir como chave estrangeira (chave primária do lado 1) na tabela do lado N do relacionamento; e
- (iii) incluir os atributos do relacionamento na relação.

Exemplo:

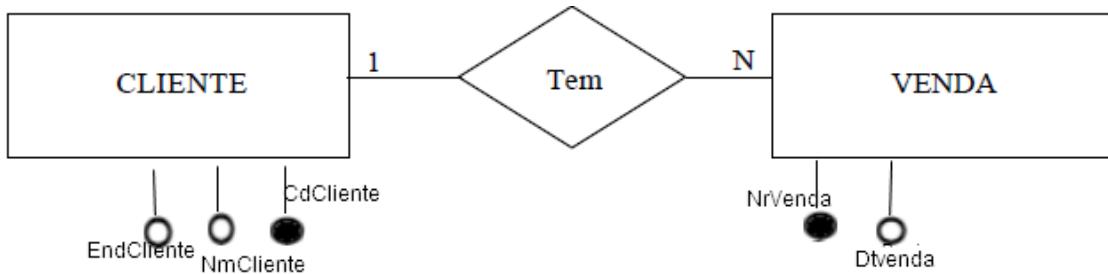


Figura 5.6. Relacionamento 1 para muitos.

A relação que representa a entidade do lado N é a Venda. Deve ser incluída a chave da relação Cliente, que representa a entidade Cliente do lado 1, como chave estrangeira na relação Venda, formando-se os seguintes esquemas relacional:

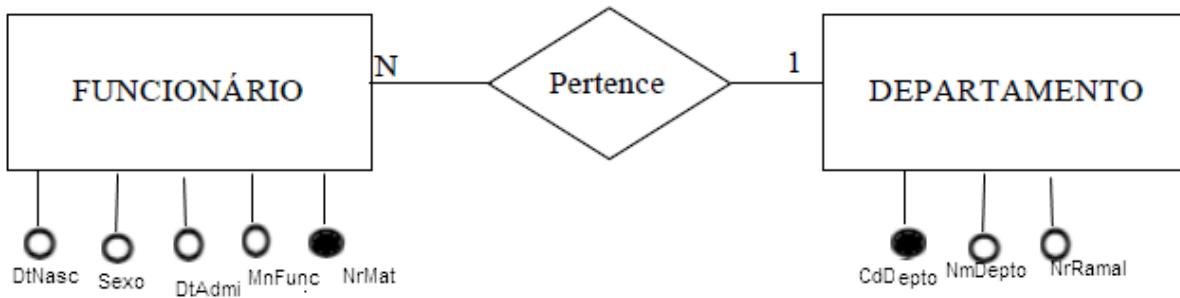
CLIENTE (**CdCliente**, **NmCliente**, **EndCliente**)

VENDA (**NrVenda**, **DtVenda**, **CdCliente**)

Início da atividade

Atividade 4 - Atende ao objetivo 2

Considere o seguinte diagrama DER:



Realize o mapeamento do diagrama acima para o modelo relacional.

Resposta comentada

Uma representação das tabelas para esse diagrama é dada por:

FUNCIONARIO (**NrMat**, **MnFunc**, **DtAdmi**, **Sexo**, **DtNasc**)

DEPARTAMENTO (CodDept, NmDept, NrRamal)

Vamos determinar como vincular uma instância de um funcionário com uma instância correspondente ao seu respectivo departamento e vice-versa.

- I. A solução é colocar na tabela Funcionários uma nova coluna 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 CdDept da tabela Departamentos para saber o nome completo do departamento ou o seu ramal.
- II. Essa coluna nova criada para implementar essa ligação 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":

FUNCIONARIO (NrMat, MnFunc, DtAdmi, Sexo, DtNasc, Dept)



DEPARTAMENTO (CodDept, NmDept, NrRamal)

Fim da atividade

Relacionamentos muitos para muitos (N:N)

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

QUEM REFERENCIA QUEM?

Nesse caso, é preciso:

- (i) criar uma nova relação para representar o relacionamento;
- (ii) incluir como chave estrangeira as chaves primárias das relações que participam do relacionamento. Essas chaves combinadas formarão a chave primária da relação;
- (iii) incluir também eventuais atributos do relacionamento.

Esse caso sempre pode ser resolvido com o desdobramento em dois relacionamentos 1 para muitos (1:N) 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 combinação das chaves primárias das duas tabelas principais.

Na Figura 5.7 temos a situação em que Fornecedor fornece muitos Produtos e qualquer Produto poderá ser fornecido por vários fornecedores.

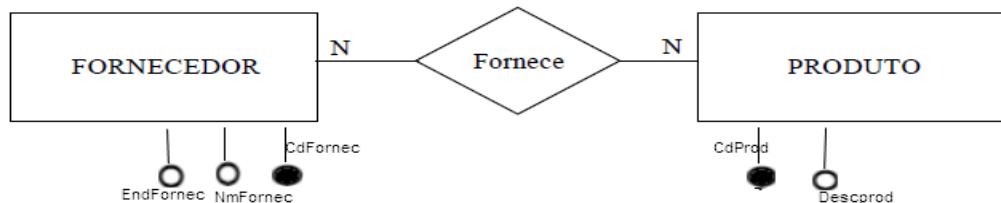


Figura 5.7: Relacionamentos muitos para muitos.

Pergunta-se então:

- * Qual tabela possuirá essa combinação 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 dessa 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 duas chaves estrangeiras (as colunas CdFornec e CdProd).

FORNECEDORES (CdFornec, NmFornec, EndFornec)

ORIGEM_PRODUTOS (CdFornec, CdProd, VrPreço, Prazo)

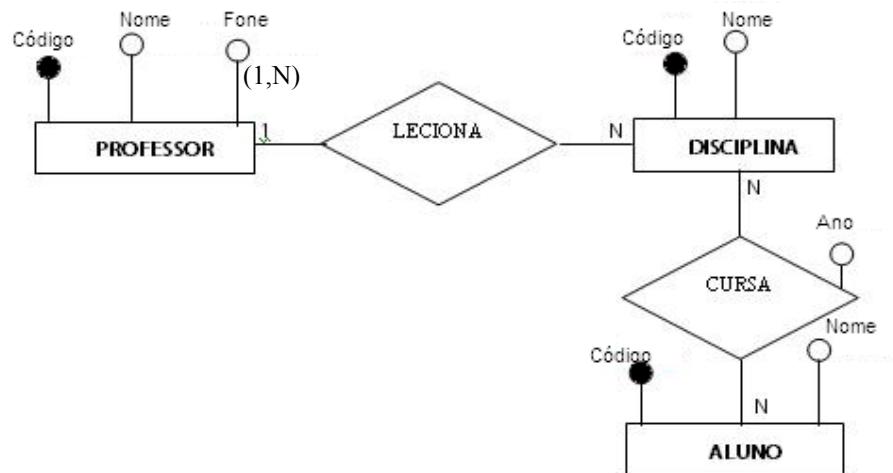
PRODUTOS (CdProd, DescProd)

Início da atividade

Atividade 5 - Atende ao objetivo 2

Vamos usar uma parte de uma aplicação simples de um sistema acadêmico que possui

professores, disciplinas e alunos dessas disciplinas. Essa aplicação está representada graficamente pela seguinte figura:



Realize o mapeamento detalhado do modelo conceitual para o modelo lógico.

Resposta comentada

Observa-se que existem dois tipos de relacionamentos: 1:N (um para muitos) e N:N (muitos para muitos). O relacionamento 1:N LECIONA, que associa um professor (entidade Professor) que leciona disciplinas (entidade Disciplina), no qual uma disciplina só pode ser lecionada por um professor. O relacionamento N:N CURSA associa vários alunos que cursam várias disciplinas. Vale ressaltar que nesse relacionamento há um atributo ano que expressa quando o aluno cursou determinada disciplina.

Implementação relacional

Uma vez mapeado para o esquema relacional, o esquema entidade/relacionamento representado na figura resultará nas seguintes tabelas:

PROFESSOR (CdProfessor, NmProfessor)

PROFESSORFONE (CdProfessor, FoneProfessor)

DISCIPLINA (CdDisciplina, NmDisciplina, CdProfessor)

ALUNO (CdAluno, NmAluno)

CURSA (CdDisciplina, CdAluno, Ano)

Descreveremos como foram aplicadas as regras de mapeamento:

REGRA 1 - ATRIBUTO

Para cada entidade no esquema E/R é preciso criar uma tabela que inclui todos os atributos não multivalorados da entidade do esquema E/R. Nesse passo, foram criadas as relações Professor, Disciplina e Aluno com os atributos correspondentes de cada entidade. Vale ressaltar que o atributo fone não foi colocado neste passo para a relação Professor, tendo em vista que é um atributo multivalorado.

REGRA 2 - Para cada relacionamento 1:N no esquema E/R:

- (i) *identificar a relação que representa a entidade do lado N;*
- (ii) *incluir como chave estrangeira a chave primária da relação que representa a entidade do lado 1; e*
- (iii) *incluir os atributos do relacionamento na relação.*

Em nosso caso, existe um relacionamento de 1:N (LECIONA) envolvendo as entidades Professor e Disciplina. A tabela que representa a entidade com cardinalidade N é a Disciplina. Deve ser incluído na tabela Disciplina um campo CodigoProfessor, que será a chave estrangeira para fazer a ligação com a tabela Professor. Se existissem atributos de relacionamentos, estes deveriam ser incluídos também na tabela Disciplina.

REGRA 3 - Para cada relacionamento N:N no esquema E/R:

- (i) *criar uma nova relação para representar o relacionamento;*
- (ii) *incluir como chave estrangeira as chaves primárias das relações que participam do relacionamento;*
- (iii) *essas chaves combinadas formarão a chave primária da relação;*
- (iv) *incluir também eventuais atributos do relacionamento.*

No nosso caso, tem-se o relacionamento CURSA, que associa as entidades Disciplina e Aluno. Para representar esse relacionamento no esquema relacional, cria-se uma relação CURSA e incluem-se nessa tabela as chaves primárias (CdDisciplina e CdAluno) das tabelas Disciplina e Aluno como chaves estrangeiras. Essas chaves

combinadas formarão a chave primária da tabela Cursa resultante. O atributo de relacionamento ano será incluído na relação resultante como chave, tendo em vista que o aluno pode fazer mais de uma vez uma disciplina.

REGRA 4 - Para cada atributo multivalorado:

- (i) *criar uma nova relação;*
- (ii) *incluir o atributo correspondente ao atributo multivalorado mais a chave primária da relação que tem esse atributo;*
- (iii) *incluir como chave primária da nova relação a combinação do atributo multivalorado mais a chave primária da entidade que tinha o atributo multivalorado.*

Nesse caso, temos o atributo multivalorado Fone. Deve ser criada uma nova tabela Fone_Professor e incluir o atributo multivalorado Fone e a chave primária da relação PROFESSOR que tem esse atributo multivalorado. A combinação do atributo multivalorado e a chave primária da relação PROFESSOR formarão a chave primária da tabela resultante.

Fim da atividade

Início da atividade online

Atividade 6 - Atende ao objetivo 2

Entre no ambiente virtual e resolva a atividade a seguir enviando sua resposta ao tutor. A partir do minimundo e do modelo conceitual apresentados a seguir, realize o mapeamento detalhado do modelo conceitual para o modelo lógico, listando tabelas, campos, chaves primárias e estrangeiras.

Minimundo 3

A empresa AutoRun é especializada no conserto de veículos automotores, incluindo carros, caminhões e motos. Devido a recentes prejuízos com o desaparecimento de peças na empresa, o gerente decidiu implantar um sistema para controlar os orçamentos emitidos e as peças utilizadas nos respectivos serviços prestados pela

empresa.

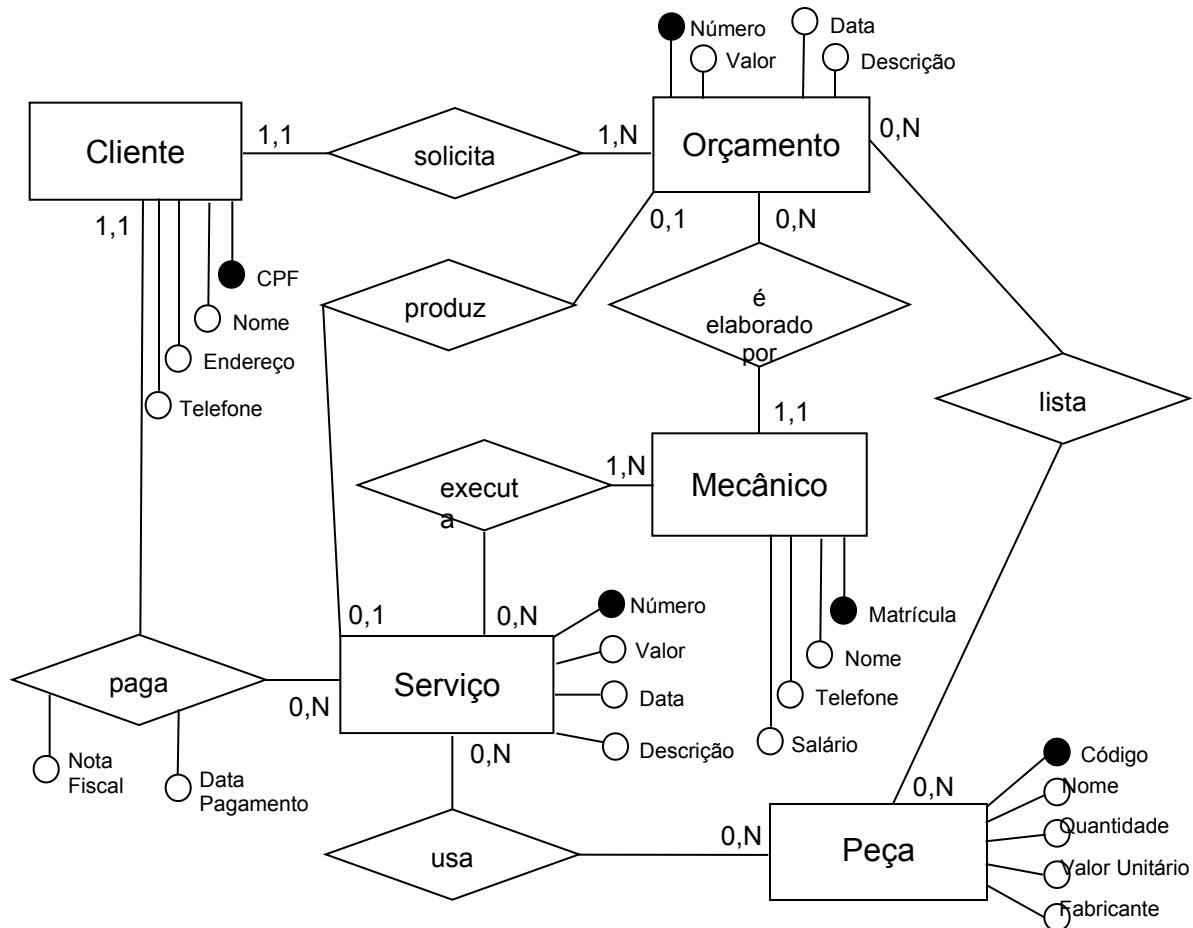
O gerente informou que, tanto no caso dos orçamentos quanto para os serviços realizados, normalmente o mecânico responsável toma nota do número do orçamento ou serviço, o valor total correspondente, a data da sua realização e acrescenta uma descrição detalhada do que foi orçado ou realizado. Ele também precisa relacionar todas as peças que foram previstas no orçamento ou utilizadas na execução do serviço, se for o caso.

No caso da solicitação de um orçamento, as seguintes informações do cliente também são registradas pelo mecânico: CPF, nome do cliente, endereço e telefone para contato. No caso de um serviço, também são anotados o número do orçamento correspondente (se houver) e os nomes de todos os mecânicos que o executaram. Ao final do serviço, é emitida uma nota fiscal que deverá ser paga pelo cliente.

O gerente informou ser imprescindível que o sistema armazene tanto o número da nota fiscal emitida para pagamento quanto a data em que esta foi paga pelo cliente. Além de controlar os orçamentos e serviços realizados pela empresa, o sistema deverá ser capaz de emitir os seguintes relatórios:

1. Listagem de peças em estoque na empresa, contendo o código da peça, o nome, o fabricante, o valor unitário e a quantidade disponível;
2. Relação mensal de gastos com os mecânicos, mostrando a matrícula, o nome, o telefone e o salário de cada um deles;
3. Listagem completa dos serviços realizados por cada mecânico no mês, exibindo o nome e a matrícula do mecânico bem como o número e o valor dos serviços executados.

Modelo conceitual produzido a partir do minimundo 3



Fim da atividade online

Mapeamento - tipos de relacionamentos binários

Cardinalidade máxima 1:1 e cardinalidade mínima 1:1

Existe a possibilidade de mapeamento dos tipos de entidades envolvidos em uma única relação, ou seja, a fusão de tabelas.

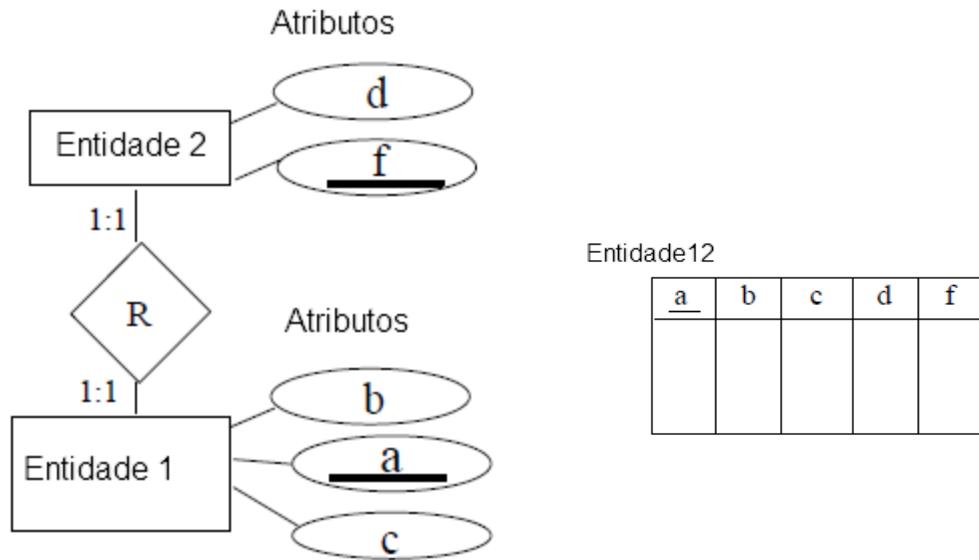
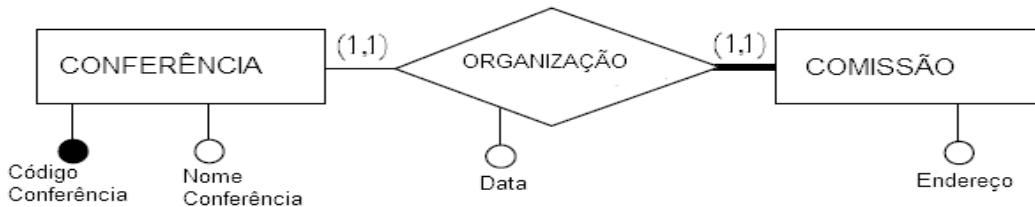


Figura 5.8: Cardinalidade máxima 1:1 e cardinalidade mínima 1:1 – esquema relacional.

Esquema relacional: ENTIDADE12 (Atributo a, Atributo b, Atributo c, Atributo d, Atributo f)

Exemplo:



Observa-se uma entidade fraca (reta em negrito), pois a identificação de suas ocorrências depende da identificação de outra(s) entidade(s), ou seja, para alguma comissão ser referenciada precisa do código de alguma conferência.

Podemos realizar a junção das entidades numa só tabela. Pode-se aplicar essa regra quando o relacionamento é de tipo 1:1. Todos os atributos de ambas as entidades, bem como os atributos do relacionamento, devem estar numa única entidade. Veja a seguir o esquema relacional correspondente:

Cardinalidade máxima 1:1 e cardinalidade mínima 0:0, 0:1 ou 1:0

Nesse caso, as entidades têm participação obrigatória. Casos desse tipo são mapeados com a colocação de um campo identificador ou da chave primária (pertencente a uma das relações envolvidas) na forma de chave estrangeira na outra relação. Preferivelmente, a relação do lado 0 (ou seja, aquela que mapeia o tipo de entidade com cardinalidade mínima 1) deve receber a chave primária da outra relação.

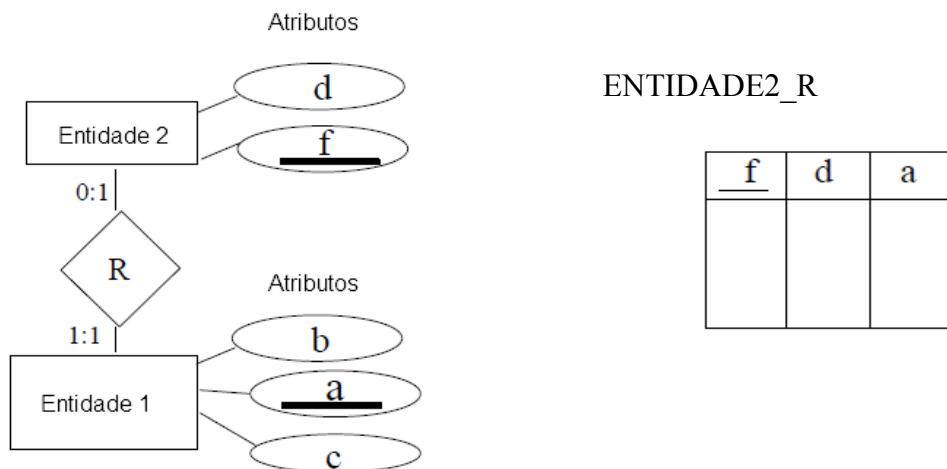
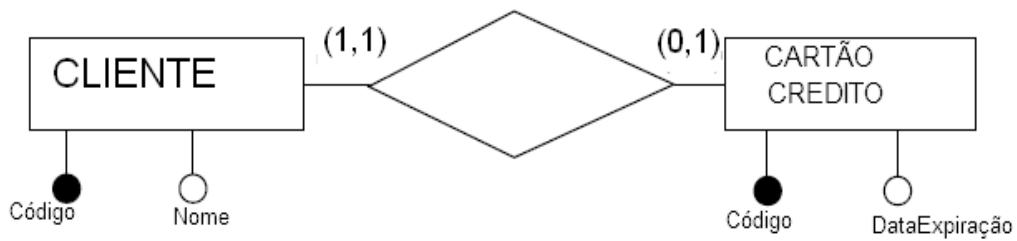


Figura 5.9: Cardinalidade máxima 1:1 e cardinalidade mínima 0:0, 0:1 ou 1:0 – esquema relacional

Entidade2_R (Atributo f, Atributo d, Atributo a)

Aplicando a regra já descrita aqui, criamos a tabela Entidade2_R, que contém ao mesmo tempo os atributos da Entidade 2 e a chave estrangeira para permitir a ligação com a Entidade 1.

Um exemplo desta situação é apresentado na figura a seguir.



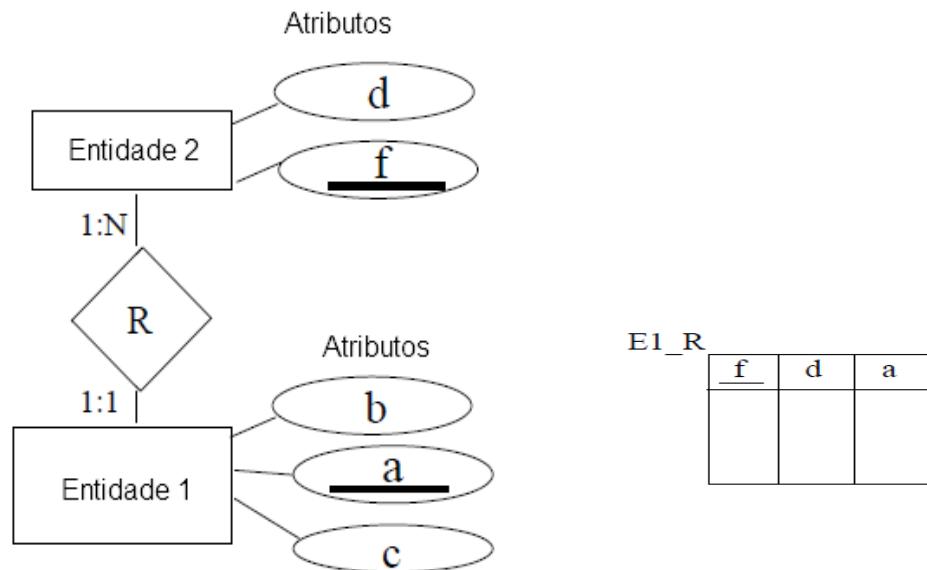
Preferivelmente, a relação Cartãocredito (lado 0) deve receber a chave primária da relação Cliente. Obtemos, assim, o seguinte esquema:

CLIENTE (CdCliente, Nome)

CARTÃOOCREDITO (CdCartão, DataExp, CdCliente)

Cardinalidade máxima 1:N ou N:1 e cardinalidade mínima 1:1, M:1 ou 1:M

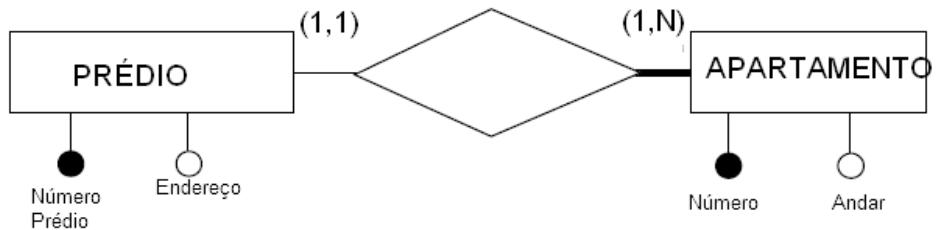
São mapeados pela colocação da chave primária de uma das relações envolvidas como chave estrangeira na outra relação. Obrigatoriamente a relação do lado N (ou seja, aquela que mapeia o tipo de entidade com cardinalidade máxima 1) deve receber a chave primária da outra relação.



Entidade1_R (Atributo f, Atributo d, Atributo a)

Figura 5.10: Cardinalidade máxima 1:N ou N:1 e cardinalidade mínima 1:1, M:1 ou 1:M
- esquema relacional

Exemplo:



Cabe observar que, neste caso, a coluna Número da tabela Apartamento (lado N) implementa o relacionamento do apartamento com seu prédio. Além de ser chave estrangeira, é também parte da chave primária. Segue o esquema relacional correspondente:

PREDIO (NoPredio, Endereco)

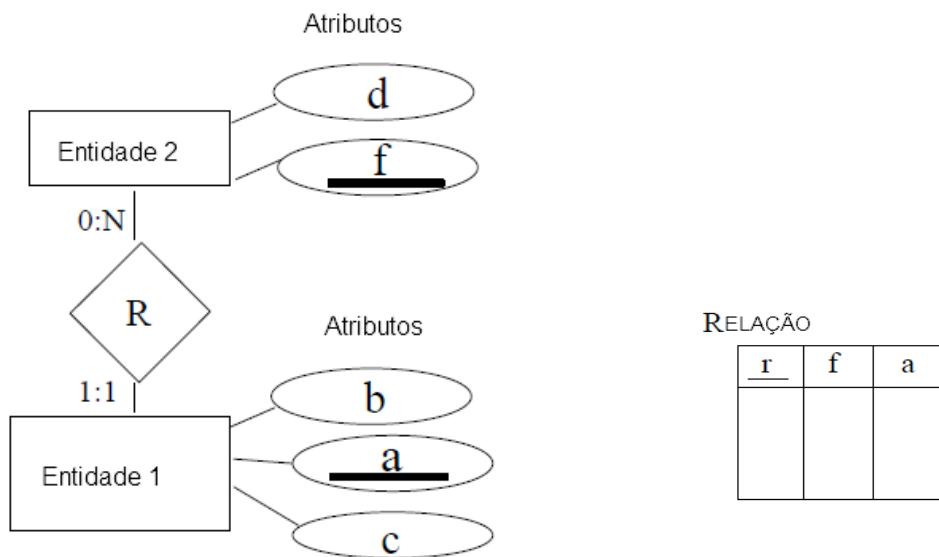


APARTAMENTO (NoPredio, NoApto, AndarAp)

No Predio referencia PREDIO

Cardinalidade máxima 1:N ou N:1 e cardinalidade mínima 0:0, 1:0, 0:1, 0:M ou M:0:

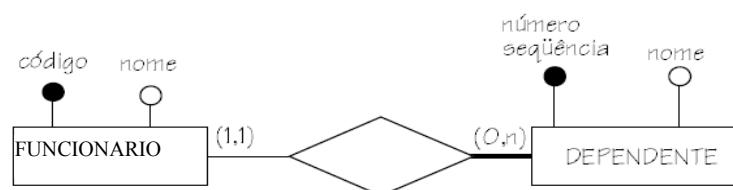
São mapeados em relações “especiais” contendo uma chave primária criada artificialmente e as chaves primárias das duas relações envolvidas. Opcionalmente, a chave primária pode ser formada pela composição das duas chaves estrangeiras. Outra forma de mapeamento é utilizar a regra anterior tendo como diferença um dos lados do tipo de relacionamento, que terá a cardinalidade máxima 0, e não 1.



RELAÇÃO (Atributo_r, Atributo_f, Atributo_a)

Figura 5.11: Cardinalidade máxima 1:N ou N:1 e cardinalidade mínima 0:0, 1:0, 0:1, 0:M ou M:0 - esquema relacional.

Exemplo: Dado o seguinte DER:



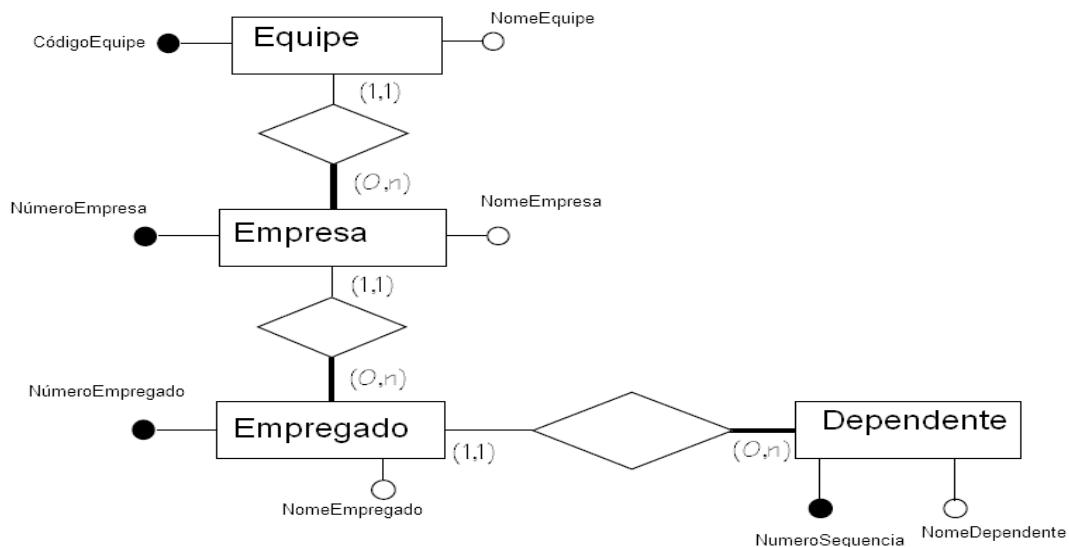
Há uma situação na qual a tradução de uma entidade para uma tabela não é trivial. Existe a situação na qual uma entidade possui um *relacionamento identificador* do tipo entidade Dependente. Um dependente é identificado pelo código do empregado ao qual ele está vinculado e por um número de sequência que distingue os diversos dependentes de um mesmo empregado.

Toda coluna que faz parte da chave primária e que é chave estrangeira corresponde a um identificador externo da entidade.

A regra de identificadores externos é que, para cada identificador externo, seja criada uma coluna (ou várias, no caso de o identificador externo ser composto de vários atributos) na tabela em questão. Essa coluna fará parte da chave primária da tabela. O esquema relacional para esse mapeamento da entidade Dependente corresponde:

DEPENDENTE (CodigoEmp, NoSeq, Nome)

Exemplo: Cabe observar que na composição da chave primária de uma tabela que possui identificador externo pode ser necessário colecionar atributos de diversas entidades, conforme mostrado na figura a seguir:



Esquema relacional correspondente:

EQUIPE (CdEquipe, NmEquipe)

EMPRESA (CdEquipe, NoEmpresa, NmEmpresa)

EMPREGADO (CdEquipe, NoEmpresa, NoEmpreg, NmEmpregado)

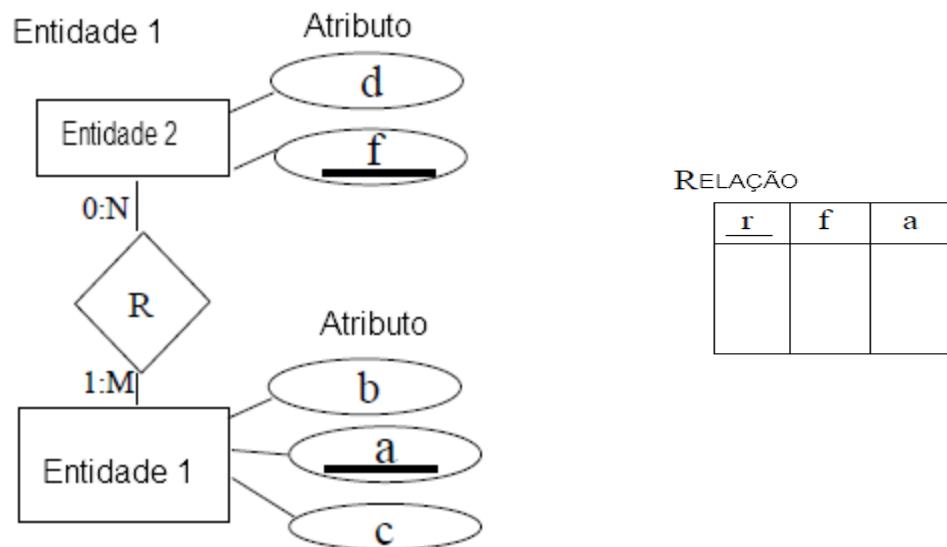
DEPENDENTE (CdEquipe, NoEmpresa, NoEmpreg, NoSeq, NmDependente)

Para compor a chave primária da tabela Dependente é necessário usar, além do atributo Número de Sequência, o identificador do empregado. Um empregado é identificado por seu número e pelo identificador da empresa à qual ele está vinculado. Por sua vez, a empresa é identificada por um número e pelo identificador da equipe ao qual ela pertence. Assim, um dependente é identificado pela combinação das seguintes informações:

- código da equipe da empresa à qual seu empregado está vinculado;
- número da empresa à qual seu empregado está vinculado;
- número de seu empregado;
- seu número de sequência; e
- nome do dependente.

Cardinalidade máxima N:N e independente da cardinalidade mínima

São mapeados em relações “especiais” contendo uma chave primária criada artificialmente e as chaves primárias das relações envolvidas. Opcionalmente, a chave primária pode ser formada pela composição das chaves estrangeiras.



RELAÇÃO (Atributo r, Atributo f, Atributo a)

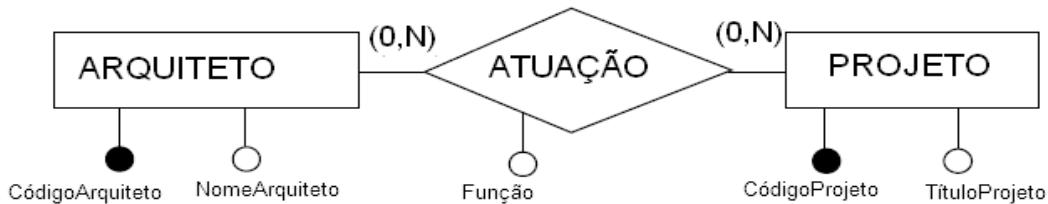
Figura 5.12: Cardinalidade máxima N:N e independente da cardinalidade mínima) – esquema relacional.

Nesse caso, foi criada uma chave primária artificial (atributo r) e colocadas as chaves estrangeiras das entidades relacionadas (atributos a e f).

Início da atividade

Atividade 7 – Atende ao objetivo 2

Considere o seguinte DER e realize o mapeamento para o modelo relacional.



Resposta comentada

A chave primária desta tabela é o conjunto das colunas correspondentes aos identificadores ou chaves das entidades relacionadas. Cada conjunto de colunas que corresponde ao identificador de uma entidade é a chave estrangeira em relação à tabela que implementa a entidade referenciada. Veja a seguir o esquema relacional correspondente:

ARQUITETO (CdArquiteto, NmArquiteto)

PROJETO (CdProjeto, TítProjeto)

ATUAÇÃO (CdArquiteto, CdProjeto, Função)

O atributo CdArquiteto referencia Arquiteto e o atributo CdProjeto referencia o Projeto.

A tabela **ATUAÇÃO** implementa o relacionamento Atuação. A chave primária da tabela é formada pelas colunas CdArquiteto e CdProjeto, que correspondem aos identificadores das entidades relacionadas Arquiteto e Projeto. Cada uma dessas colunas são chaves estrangeiras das tabelas que implementam a entidade relacionada.

A coluna Função corresponde ao atributo do relacionamento.

Fim da atividade

Início da atividade online

Atividade 8 - Atende ao objetivo 2

Entre no ambiente virtual e resolva a atividade a seguir enviando sua resposta ao tutor.

A partir do minimundo e do modelo conceitual apresentados à frente, realize o mapeamento detalhado do modelo conceitual para o modelo lógico, listando as tabelas, campos, chaves primárias e estrangeiras.

Minimundo 5

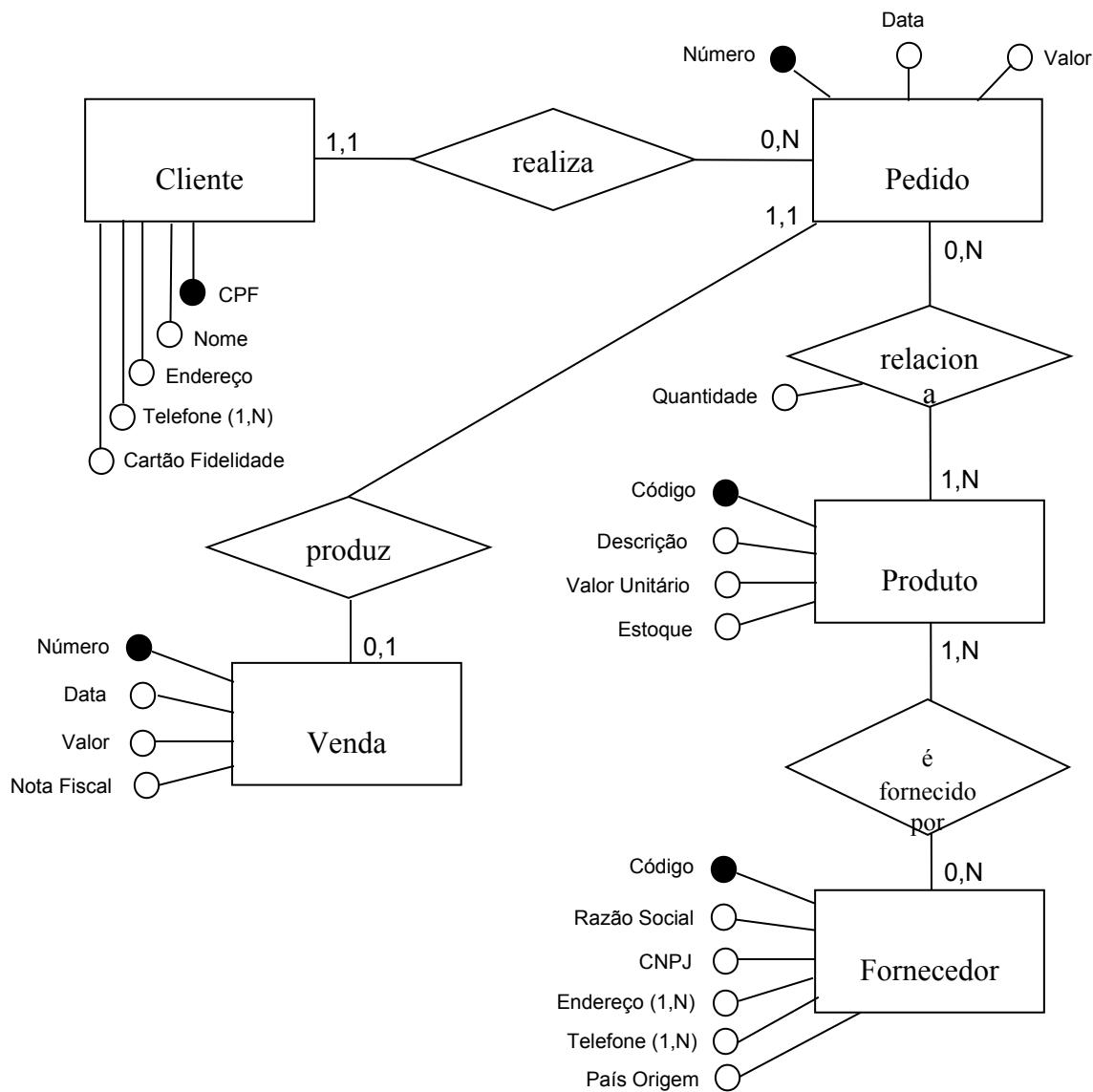
1. A empresa “Importex” é especializada na importação e venda local de produtos originados de outros países.
2. A empresa tem perdido várias vendas devido à falta do produto em estoque quando o cliente faz um pedido para comprá-lo. A diretoria da empresa decidiu, então, implantar um sistema que permita informar o nível de estoque dos produtos e monitorar as vendas para evitar que incorra em novos prejuízos.
3. O usuário designado pela diretoria informou que, quando o cliente solicita um ou mais produtos, o funcionário verifica se os mesmos existem em estoque. Em caso negativo, ele anota os produtos que devem ser comprados e sua respectiva quantidade de reposição. Se os produtos estiverem disponíveis, o atendente encaminha ao setor de vendas um pedido especificando os produtos e as quantidades solicitadas pelo cliente. Todo pedido possui um número único, que o identifica. Os produtos são, então, separados e embalados. Simultaneamente, o setor de vendas emite uma nota fiscal que relaciona os produtos adquiridos, respectivas quantidades e preços de venda. O cliente efetua o pagamento no valor total da nota fiscal. Clientes com cartão de fidelidade têm direito a 10% de desconto sobre o total da nota.
4. O sistema a ser desenvolvido deve registrar cada venda efetuada, informando o número da venda (originado do pedido), o número da nota fiscal, a data e o valor total. A fim de saber se o cliente tem direito a desconto, a empresa mantém um cadastro de

clientes onde constam CPF, endereço, nome, número do cartão de fidelidade e seus telefones. Da mesma forma, um cadastro de fornecedores lista todas as informações necessárias para contato no caso de aquisição de novos produtos.

5. Com o objetivo de evitar que novas vendas sejam perdidas devido à falta de produto em estoque, o sistema deverá ser capaz de emitir os seguintes relatórios:

- a. Listagem de pedidos pendentes relacionando para cada um o seu número, data e valor;
- b. Listagem de produtos sem estoque disponível, contendo código do produto, descrição, valor unitário;
- c. Relatório das vendas diárias dos produtos, relacionando o número da nota fiscal, código e descrição do produto, data e valor da nota fiscal;
- d. Listagem dos fornecedores de produtos, ordenada por país de origem. A listagem deve exibir o código, a razão social, o(s) endereço(s) e o(s) telefone(s) para contato.

Modelo conceitual produzido a partir do minimundo 5



Fim da atividade online

Relacionamentos de grau maior que 2

As regras apresentadas até este ponto se aplicam somente à implementação de relacionamentos binários, isto é, que envolvem apenas duas entidades. Para relacionamentos de grau maior que dois não são definidas regras específicas.

Cardinalidade máxima M:N:P e independente da cardinalidade mínima

Em geral, a implementação de um relacionamento de grau maior que dois dá-se na seguinte sequência de passos:

1. O relacionamento é transformado em uma entidade. Essa nova entidade é ligada, por um relacionamento binário, a cada uma das entidades que participavam dos relacionamentos originais.
2. As regras de implementação de entidades e relacionamentos binários apresentadas nesta aula são então aplicadas às entidades e aos relacionamentos binários que foram assim criados.

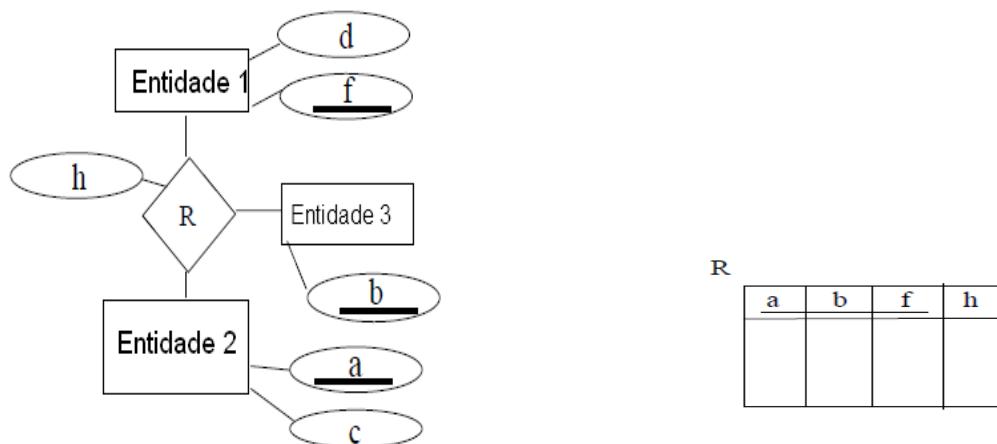
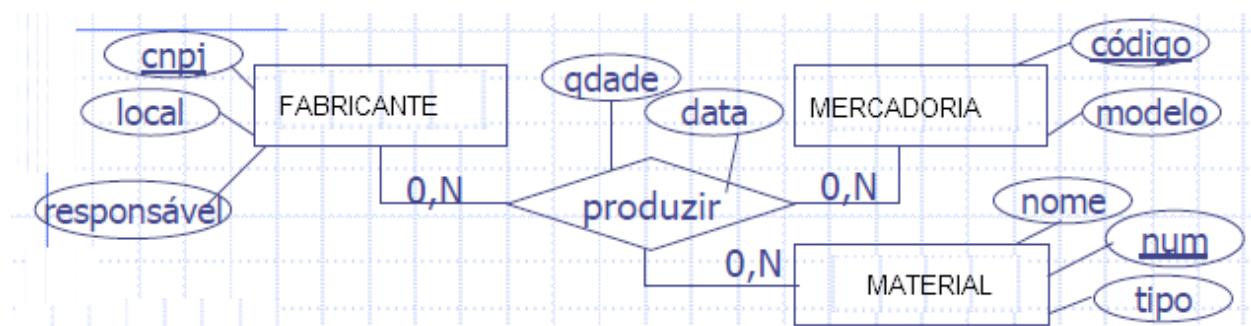
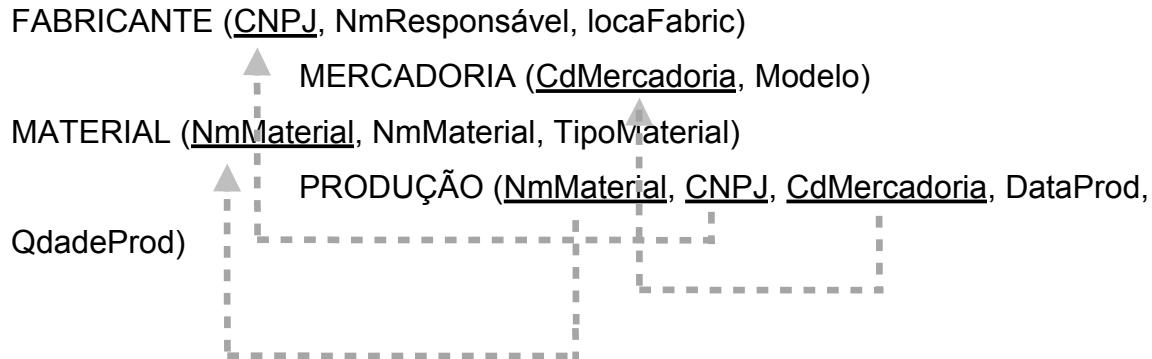


Figura 5.13: Cardinalidade máxima M:N:P e independente da cardinalidade mínima.

Exemplo:



Temos os seguintes esquemas resultantes do mapeamento para o modelo relacional:



Conclusão

Nesta aula, você estudou como mapear um modelo conceitual transformando-o em um modelo relacional. Para tal, utilizam-se diversas regras apresentadas para os relacionamentos binários demonstrados, fazendo uso de conceitos básicos do modelo relacional, tais como atributos e chaves primária e estrangeira, entre outros.

Resumo

Em um ambiente de banco de dados relacional, utilizamos alguns conceitos muito importantes.

- Modelo de dados relacional: todos os dados estão guardados em tabelas ou relações. Toda a sua definição é teórica e baseada em lógica, álgebra relacional e na teoria de conjuntos.

Definem-se:

1. Relação ou tabela:
 - É a estrutura que armazena os dados referentes a uma instância. Uma tabela é composta por colunas e linhas.
2. Linhas ou tuplas: cada linha representa uma coleção de dados relacionados.
3. Coluna ou atributo:
 - Cada tipo de informação armazenada numa tabela é uma coluna. Toda coluna de uma tabela deve possuir um nome pelo qual será referenciada.
4. Domínio: conjunto de valores possíveis em cada coluna.

Tipos de chaves

1. Chave primária:

- O atributo identificador que não se repete entre as linhas de uma tabela é chamado de chave primária.

2. Chaves compostas:

- Formadas pela combinação de vários atributos de chaves primárias.

3. Chave candidata:

- Numa entidade que contenha mais de uma chave primária existirá sempre ao menos uma chave candidata.

4. Chave alternativa ou secundária:

- É a chave candidata que não foi escolhida como chave primária.

5. Chave estrangeira:

- É uma coluna (ou combinação de colunas) que indica um valor que deve existir como chave primária numa outra tabela.

Conceito de nulo

- Representa um valor desconhecido para um dado. Não se deve confundir o conceito de nulo com espaços em branco ou o número zero.

Regras básicas de mapeamento de um modelo ER em um modelo relacional

Dá-se nas seguintes transformações:

1. Entidades e respectivos atributos:

- Cada entidade ou relacionamento com atributos próprios deve corresponder a uma tabela ou relação do banco de dados;
- Cada atributo da entidade define uma coluna dessa tabela;
- Os atributos identificadores da entidade correspondem às colunas que compõem a chave primária da tabela.

2. Relacionamentos e atributos

- 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;
- 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 multivalorados.

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 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. Dicas práticas:

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

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.

- Identificar a relação que representa a entidade do lado N;
- Incluir como chave estrangeira (chave primária do lado 1) na tabela do lado N do relacionamento; e
- Incluir os atributos do relacionamento na relação.

Relacionamentos muitos para muitos

Este caso pode ser resolvido sempre com o desdobramento em dois relacionamentos de 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.

- Criar uma nova relação para representar o relacionamento;

- Incluir como chave estrangeira as chaves primárias das relações que participam do relacionamento. Essas chaves combinadas formarão a chave primária da relação;
- Incluir também eventuais atributos do relacionamento.

Mapeamento - tipos de relacionamentos binários

Cardinalidade máxima 1:1 e cardinalidade mínima 1:1

Existe a possibilidade de mapeamento dos tipos de entidades envolvidos em uma única relação, ou seja, fusão de tabelas.

Cardinalidade máxima 1:1 e cardinalidade mínima 0:0, 0:1 ou 1:0

As entidades têm participação obrigatória. São mapeadas através da colocação do identificador ou da chave primária (pertencente a uma das relações envolvidas) na forma de chave estrangeira na outra relação. Preferivelmente, a relação do lado 0 (ou seja, aquela que mapeia o tipo de entidade com cardinalidade mínima 1) deve receber a chave primária da outra relação.

Cardinalidade máxima 1:N ou N:1 e cardinalidade mínima 1:1, M:1 ou 1:M

São mapeados pela colocação da chave primária de uma das relações envolvidas como chave estrangeira na outra relação. Obrigatoriamente, a relação do lado N (ou seja, aquela que mapeia o tipo de entidade com cardinalidade máxima 1) deve receber a chave primária da outra relação.

Cardinalidade máxima 1:N ou N:1 e cardinalidade mínima 0:0, 1:0, 0:1, 0:M ou M:0

São mapeados em relações “especiais” contendo uma chave primária criada artificialmente e as chaves primárias das duas relações envolvidas. Opcionalmente, a chave primária pode ser formada pela composição das duas chaves estrangeiras. Outra forma de mapeamento é utilizar a regra anterior tendo como diferença um dos lados do tipo de relacionamento, que terá a cardinalidade máxima 0 e não 1.

Cardinalidade máxima M:N e independente da cardinalidade mínima

São mapeados em relações “especiais” contendo uma chave primária criada artificialmente e as chaves primárias das relações envolvidas. Opcionalmente, a chave primária pode ser formada pela composição das chaves estrangeiras.

Relacionamentos de grau maior que 2

Cardinalidade máxima M:N:P e independente da cardinalidade mínima

Em geral, a implementação de um relacionamento de grau maior que dois dá-se na seguinte sequência de passos:

1. O relacionamento é transformado em uma entidade. Essa nova entidade é ligada, por meio de um relacionamento binário, a cada uma das entidades que participavam do relacionamento original.
2. São mapeados em relações “especiais” contendo uma chave primária criada artificialmente e as chaves primárias das relações envolvidas. Opcionalmente, a chave primária pode ser formada pela composição das chaves estrangeiras.

Informações sobre a próxima aula

Na próxima aula, você verá como instalar e configurar a ferramenta MySQL, que será utilizada para converter o modelo relacional de banco de dados em modelo físico.

Referências bibliográficas

DATE, C.J. *Introdução a Sistemas de Bancos de Dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.

CARVALHO, C.R. *SQL - Guia Prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.

ELMASRI, R.; NAVATHE, S. *Sistemas de Banco de Dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.

HEUSER, C.A. *Projeto de Banco de Dados*. 4^a ed. Porto Alegre: Bookman, 2009.

SETZER, V.W.; CORRÊA DA SILVA, F. S. 2005. *Bancos de Dados*. São Paulo: Edgard Blucher, 2005.

SILBERSCHATZ, A.; KORTH, H. *Sistema de Banco de Dados*. 3^a ed. São Paulo: Pearson Makron Books, 2008.

Disciplina de Extensão: Modelando e implementando bancos de dados relacionais
Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 6

Instalação da ferramenta MySQL

Meta

Apresentar a sequência de passos para instalar e configurar a ferramenta MySQL, necessária para a implementação do projeto físico dos bancos de dados a partir de modelos relacionais.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Instalar a ferramenta MySQL;
2. Configurar a ferramenta MySQL para permitir a criação de bases de dados e seus objetos.

Pré-requisitos

Para o correto e completo aprendizado desta aula, é importante você relembrar:

- Como produzir diagramas ER (entidade-relacionamento), conceito visto nas aulas 2, 3 e 4;
- Como converter diagramas ER em modelos relacionais, conceito visto na aula 5.

Rumo ao projeto físico

Agora que você já viu como produzir um modelo relacional a partir de um diagrama ER (entidade-relacionamento), a próxima etapa é aprender como instalar e configurar corretamente a ferramenta MySQL. Assim, você poderá implementar o projeto físico das bases de dados que idealizou.

Fim da introdução

Etapa 1: Baixando o arquivo de instalação da ferramenta MySQL

Antes de iniciar a instalação da ferramenta, é necessário baixar o arquivo de instalação. Para tal, execute os seguintes passos:

Passo 1: Acesse o link <http://www.mysql.com/downloads/mysql/>. Após acessar o link, será necessário também selecionar a plataforma utilizada pelo seu computador, caso o sistema operacional instalado seja diferente do Microsoft Windows. Se for o caso, basta trocar a opção na caixa de seleção da plataforma, conforme pode ser visualizado na Figura 6.1.

The screenshot shows the MySQL Community Server 5.5.10 download page. At the top left, there is a button labeled "Generally Available (GA) Releases". Below it, the title "MySQL Community Server 5.5.10" is displayed. Underneath the title, there is a "Select Platform:" label followed by a dropdown menu containing the option "Microsoft Windows". To the right of the dropdown menu is a red arrow pointing towards it. Below the dropdown, there is a table listing five download options, each with a "Download" button. The options are: "Windows (x86, 32-bit), MSI Installer" (mysql-5.5.10-win32.msi), "Windows (x86, 64-bit), MSI Installer" (mysql-5.5.10-winx64.msi), "Windows (x86, 32-bit), ZIP Archive" (mysql-5.5.10.zip), "Windows (x86, 32-bit), ZIP Archive" (mysql-5.5.10-win32.zip), and "Windows (x86, 64-bit), ZIP Archive" (mysql-5.5.10-winx64.zip). Each row includes file size and MD5 checksum information.

| Platform | File Type | Version | Size | Action |
|---------------------------|---------------|---------|--------|--|
| Windows (x86, 32-bit) | MSI Installer | 5.5.10 | 121.6M | Download |
| (mysql-5.5.10-win32.msi) | | | | MD5: 9773001c1c30693fc8c9b143b8846c6 Signature |
| Windows (x86, 64-bit) | MSI Installer | 5.5.10 | 123.7M | Download |
| (mysql-5.5.10-winx64.msi) | | | | MD5: 2aec658dca86129cf8c4d420bb48ce1 Signature |
| Windows (x86, 32-bit) | ZIP Archive | 5.5.10 | 27.1M | Download |
| (mysql-5.5.10.zip) | | | | MD5: 1ae2df8ee63b6a5140ae2d490e0f110 Signature |
| Windows (x86, 32-bit) | ZIP Archive | 5.5.10 | 132.9M | Download |
| (mysql-5.5.10-win32.zip) | | | | MD5: 2f8c0857dalflf8a16eb47a921554885 Signature |
| Windows (x86, 64-bit) | ZIP Archive | 5.5.10 | 135.2M | Download |
| (mysql-5.5.10-winx64.zip) | | | | MD5: 3dd0cccd63f0cbe0616feb4199d8e2fc8 Signature |

Figura 6.1: Selecionando a plataforma para baixar o arquivo de instalação do MySQL.

Passo 2: Verifique se o sistema operacional instalado no seu computador é de 32 bits ou 64 bits. Se você estiver com o Windows instalado, você pode fazer isso acessando a opção Sistema e Segurança / Sistema no painel de controle. A informação pode ser visualizada no item Tipo de sistema, como apresentado na Figura 6.2:

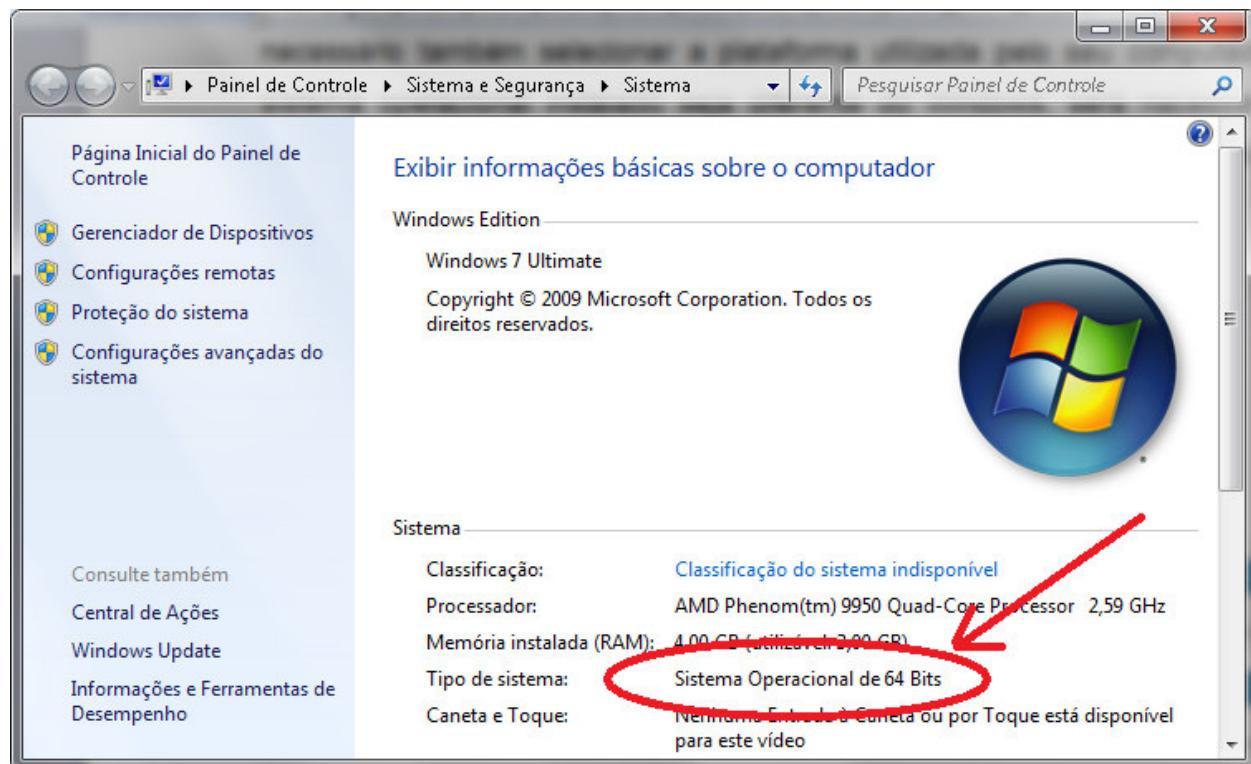


Figura 6.2: Visualizando o tipo de sistema operacional instalado.

Passo 3: Pronto. Agora que você já descobriu qual tipo de sistema operacional (32 bits ou 64 bits) está instalado no seu computador, clique no botão *Download* referente ao arquivo *MSI Installer* correspondente (um dos primeiros da lista). Os botões de download podem ser vistos na Figura 6.1.

Passo 4: Após clicar no botão *Download*, aparecerá uma página semelhante à da Figura 6.3. Essa tela pede para você se identificar antes de baixar o arquivo, mas você pode pular esta etapa. Para tal, clique no link *No thanks, just take me to the downloads!*, que é exibido ao final da página.

Select a Mirror to Start Downloading - mysql-5.5.10-win

Please take the time to let us know about you.

If this is the first time you have downloaded from us, you will be sent a password to enable you to log in.

If you already have a MySQL.com account, save time by logging in now.

Returning Users

Save time by logging in

Email:

Password:

[Forgot your password?](#)

Login

New Users

[Proceed with registration](#)

Proceed

» [No thanks, just take me to the downloads!](#)

Figura 6.3: Pulando a etapa de identificação do usuário.

Passo 5: Escolha um dos servidores para baixar o arquivo de instalação da ferramenta MySQL. Observe a Figura 6.4. Repare que os servidores estão agrupados por região/país. Procure um dos servidores listados para o Brasil e clique no link HTTP correspondente ao servidor que você escolheu para começar a baixar o arquivo.

You are downloading:

mysql-5.5.10-winx64.msi

To make this download faster, please choose a mirror site close to you from the list below.

Mirrors in: Brazil

We have looked up your IP address using [MaxMind GeoIP](#), and believe that these mirrors may be

-  Linorg - USP - Sao Paulo [HTTP](#) [FTP](#)
-  Universidade de Sao Paulo - CCE [HTTP](#) [FTP](#)

Europe

| | |
|--|--|
|  Kangaroot Linux Solutions, Belgium | HTTP FTP |
|  Easynet, Belgium | HTTP FTP |
|  Masaryk University in Brno, Czech Republic | HTTP FTP |

Figura 6.4: Selecionando o servidor para baixar o arquivo de instalação do MySQL.

Passo 6: Surgirá então uma janela *pop-up* solicitando a sua confirmação para baixar o arquivo de instalação da ferramenta MySQL (veja a Figura 6.5). Clique no botão *Download* para confirmar que você deseja baixar o arquivo.

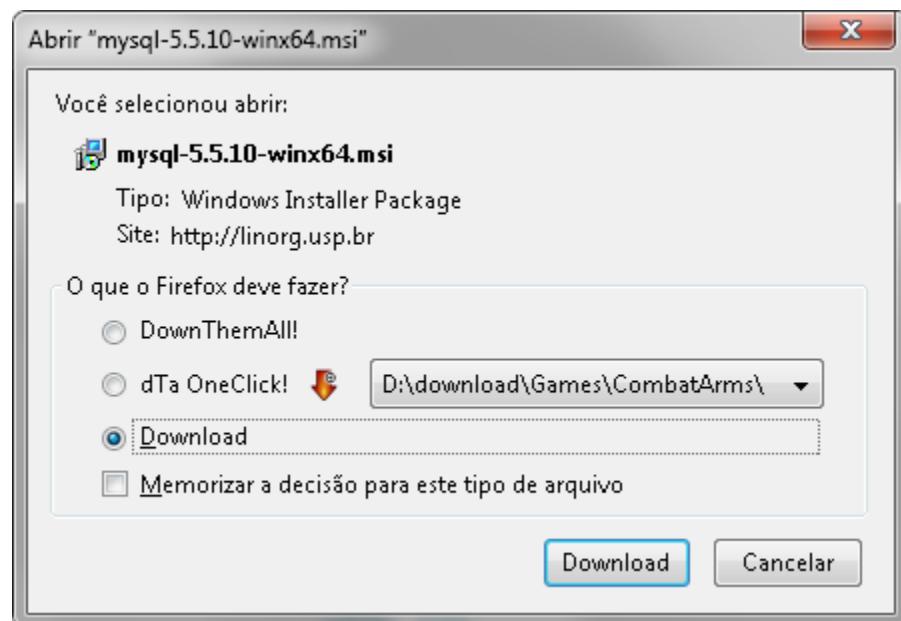


Figura 6.5: Janela *pop-up* para confirmação de *download* do arquivo no navegador Firefox.

Etapa 2: Instalação da ferramenta MySQL

Muito bem! Agora que você já terminou de baixar o arquivo de instalação da ferramenta MySQL, clique duas vezes sobre ele para iniciar o processo de instalação.

Passo 1: Será exibida uma janela *pop-up* de boas-vindas, semelhante à apresentada na Figura 6.6. Clique no botão *Next* para prosseguir.



Figura 6.6 – Janela *pop-up* de boas-vindas do instalador da ferramenta MySQL

Passo 2: Aparecerá um termo de licença de uso da ferramenta MySQL. Clique na caixa *I accept the terms in License Agreement* (Figura 6.7) para aceitar os termos da licença de uso e clique novamente no botão *Next*;

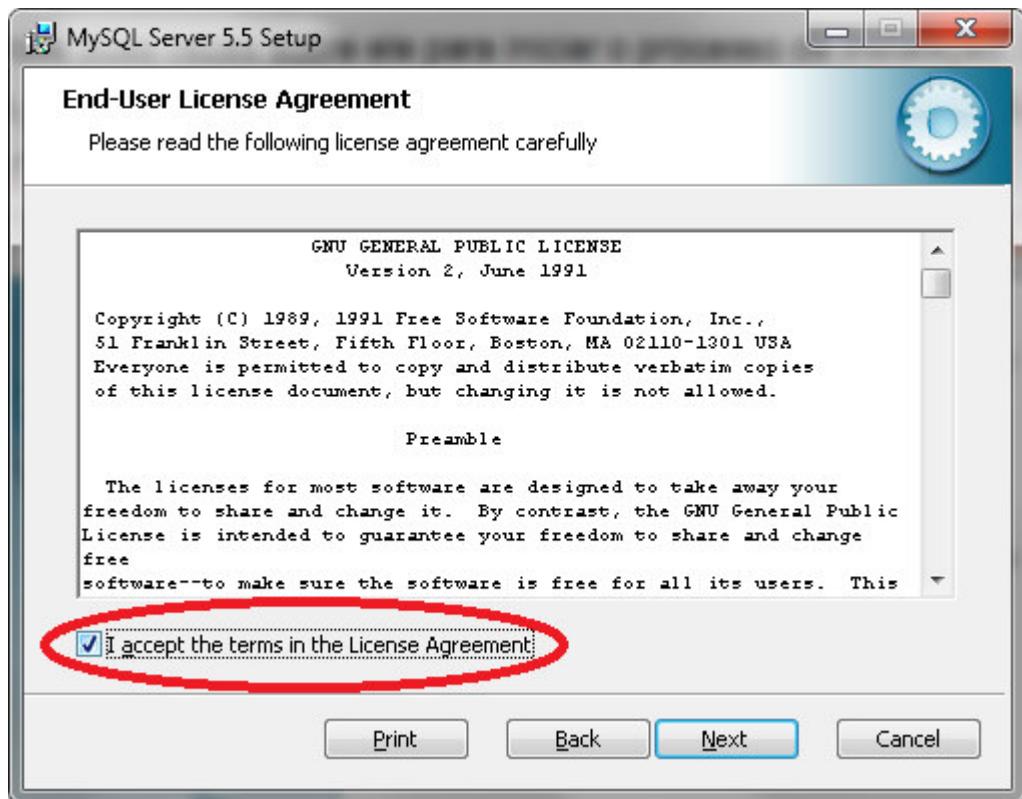


Figura 6.7: Aceitando os termos de uso da licença de instalação da ferramenta MySQL.

Passo 3: Aparecerá uma tela para a escolha do modo de instalação a ser realizado no seu computador, como pode ser visto na Figura 6.8. O modo típico é o recomendado para quem está instalando a ferramenta MySQL pela primeira vez. Os outros modos apresentam opções de instalação mais avançadas que permitem ajustar diversos detalhes durante o processo de instalação. Clique no botão *Typical* para selecionar o modo de instalação típico.

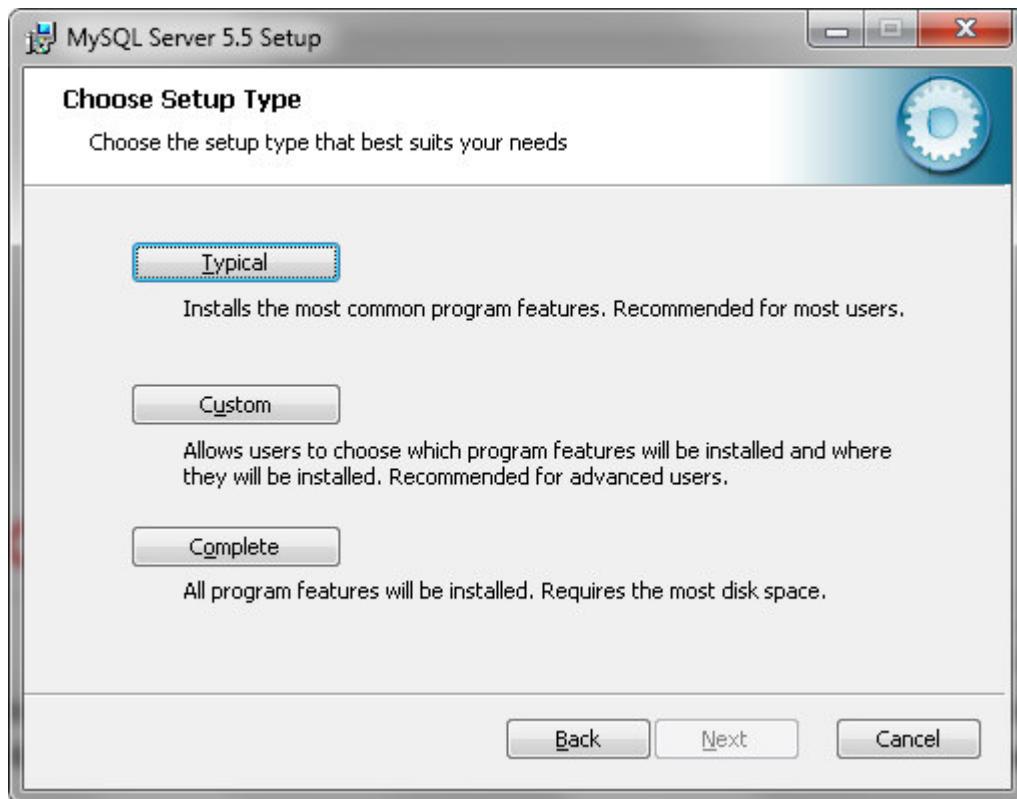


Figura 6.8: Seleção do modo de instalação da ferramenta MySQL.

Passo 4: Surgirá uma tela para confirmar o início da instalação da ferramenta MySQL, como mostra a Figura 6.9. Clique no botão *Install* para iniciar a instalação.

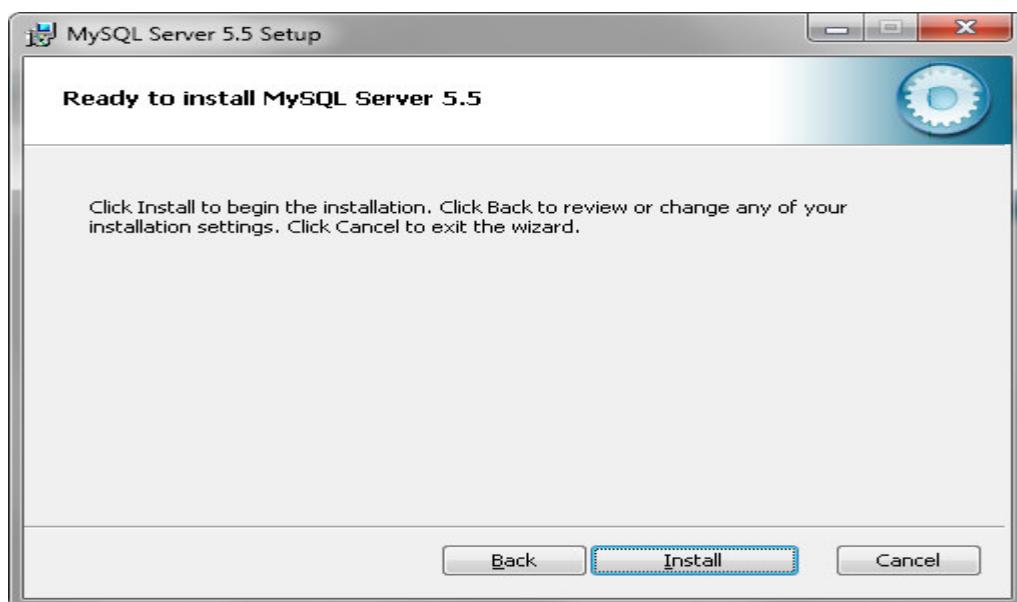


Figura 6.9: Iniciação do processo típico de instalação da ferramenta MySQL.

Passo 5: A instalação será iniciada e uma barra de *status* mostrará seu progresso (Figura 6.10). Em seguida, surgirá uma nova janela *pop-up* que exibe propaganda do serviço de suporte à ferramenta MySQL (Figura 6.11). Clique no botão *Next* nesta tela e na tela seguinte para prosseguir.

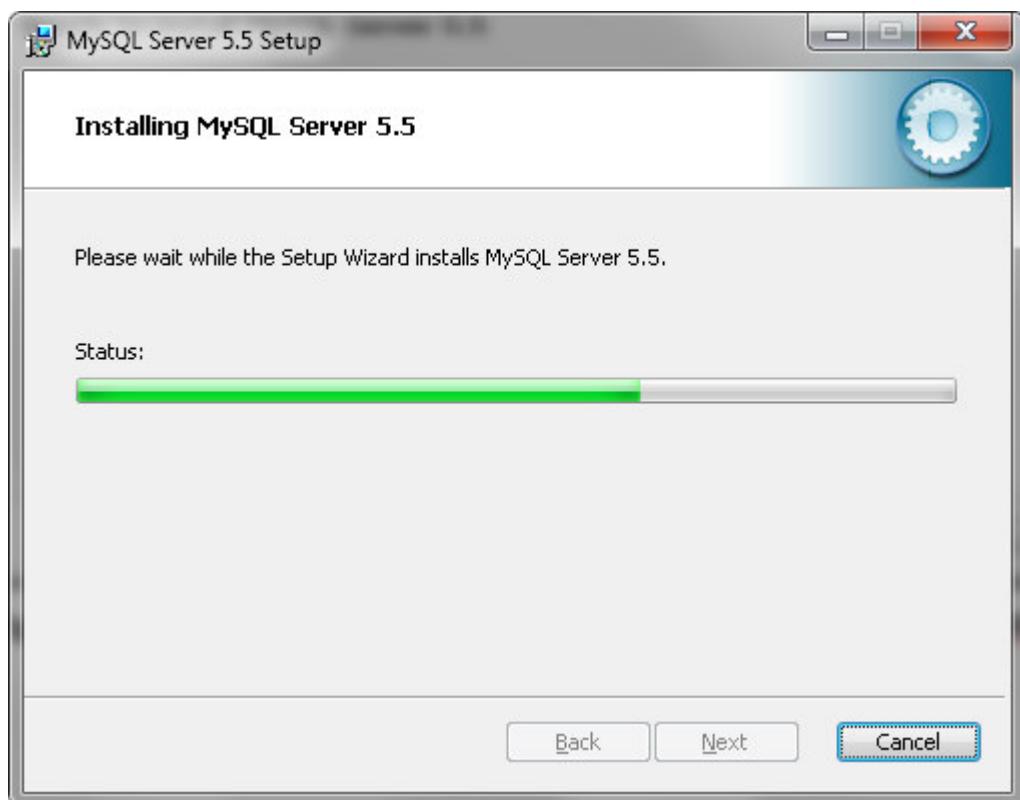


Figura 6.10: *Status* de progresso da instalação da ferramenta MySQL.



Figura 6.11: Tela de propaganda do suporte à ferramenta MySQL.

Passo 6: Ao final da instalação, será exibida uma tela semelhante à Figura 6.12. A tela indica que o processo de instalação foi finalizado e oferece uma opção para ativar o programa de configuração da ferramenta MySQL. Marque a opção *Launch the MySQL instance configuration wizard*. Clique no botão *Finish* para iniciar o processo de configuração da ferramenta.

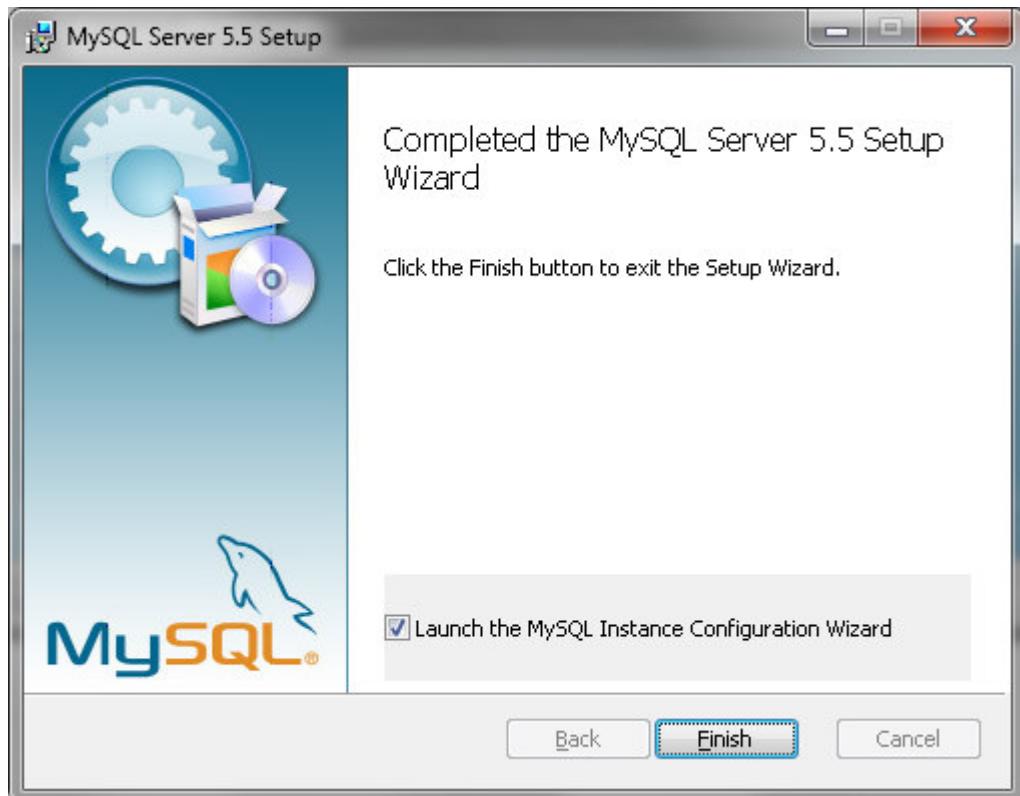


Figura 6.12: Finalização da instalação da ferramenta MySQL.

Início da atividade

Atividade Online 1 - Atende ao objetivo 1

Agora que você já aprendeu como baixar o arquivo de instalação da ferramenta MySQL e viu os passos da instalação, realize a instalação da ferramenta no seu computador.

Fim da atividade

Etapa 3: Configuração da ferramenta MySQL

Agora estamos chegando ao fim. Você já baixou e instalou a ferramenta MySQL. Vamos partir para a configuração das opções existentes. Para tal, siga os passos descritos:

Passo 1: Após a instalação, será exibida uma nova janela *pop-up* de configuração do servidor MySQL (veja Figura 6.13). Não se preocupe; clique no botão *Next* para iniciar o processo de configuração da ferramenta.

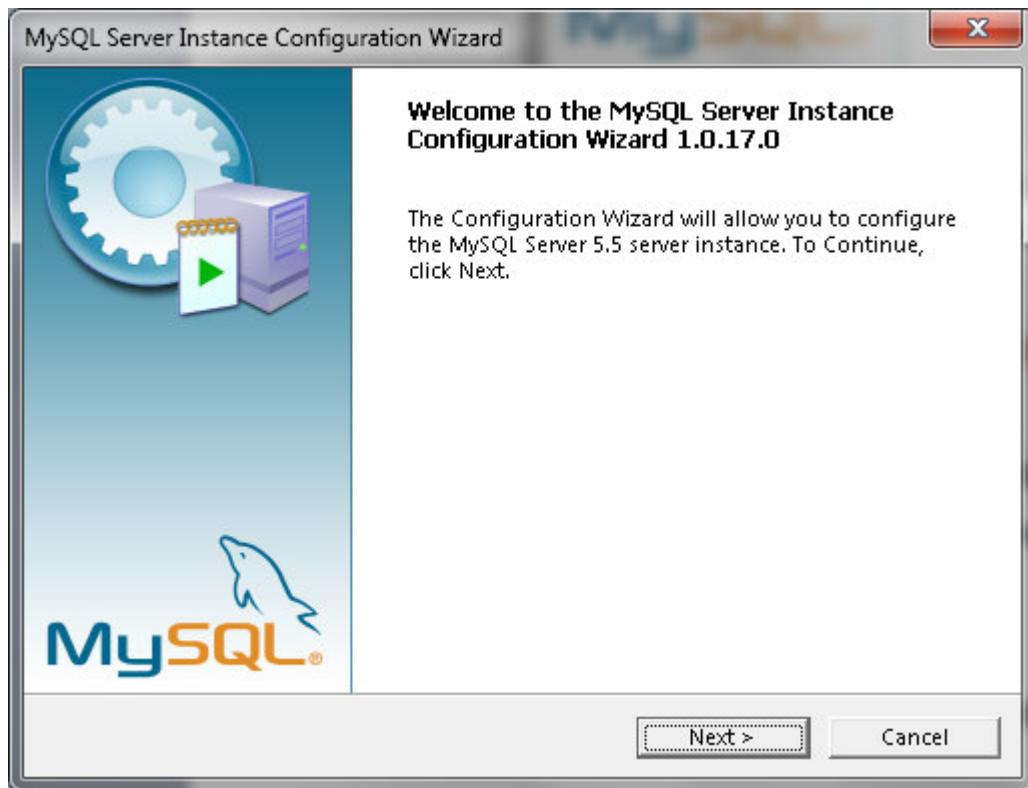


Figura 6.13: Janela *pop-up* de boas-vindas à configuração da ferramenta MySQL.

Passo 2: Aparecerá agora a tela disposta na Figura 6.14, que exibe opções de configuração do servidor de banco de dados MySQL. A ferramenta MySQL permite que a configuração seja realizada de forma detalhada ou na opção padrão. Mantenha selecionada a opção de configuração detalhada (*Detailed Configuration*) e clique no botão *Next*.

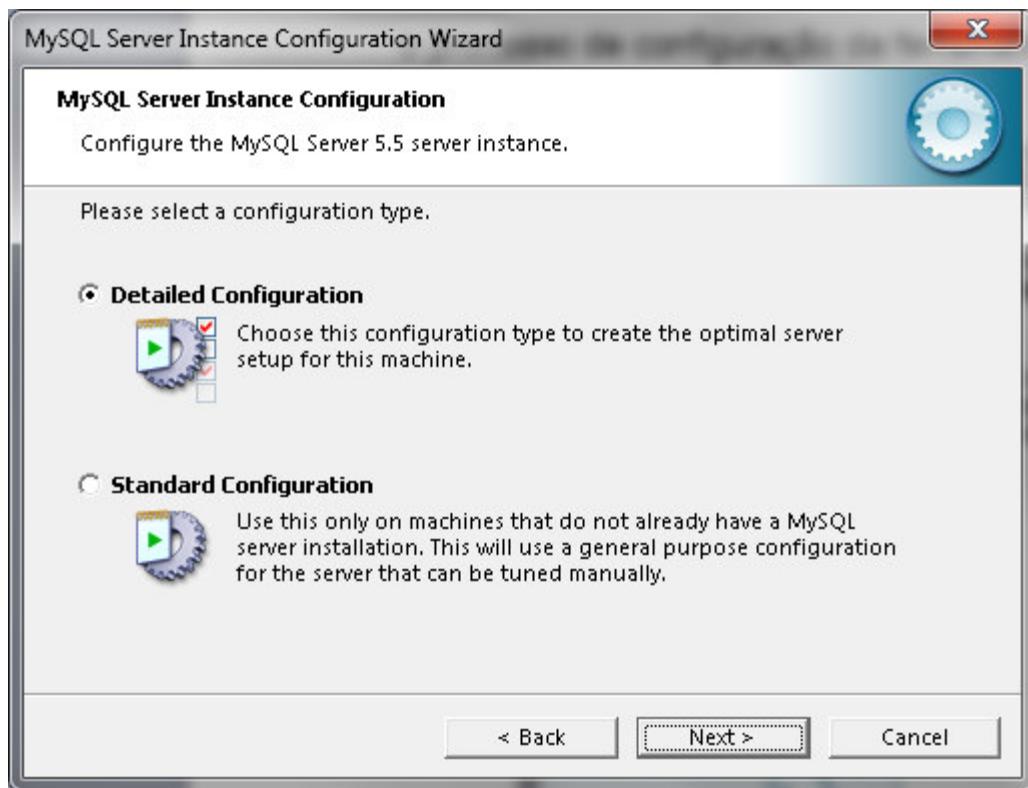


Figura 6.14 – Opções de configuração do servidor MySQL

Passo 3: Surgirá uma tela para a escolha do tipo de servidor MySQL (Figura 6.15). As opções *Server Machine* e *Dedicated MySQL Server Machine* consomem mais memória do computador e servem para criar um servidor MySQL (dedicado ou não) que pode ser acessado por outras máquinas na rede. A opção “Máquina de desenvolvedor” (*Developer machine*) é a mais simples e a que consome menos memória na criação do servidor MySQL. Selecione esta opção e clique em *Next* para seguir adiante.

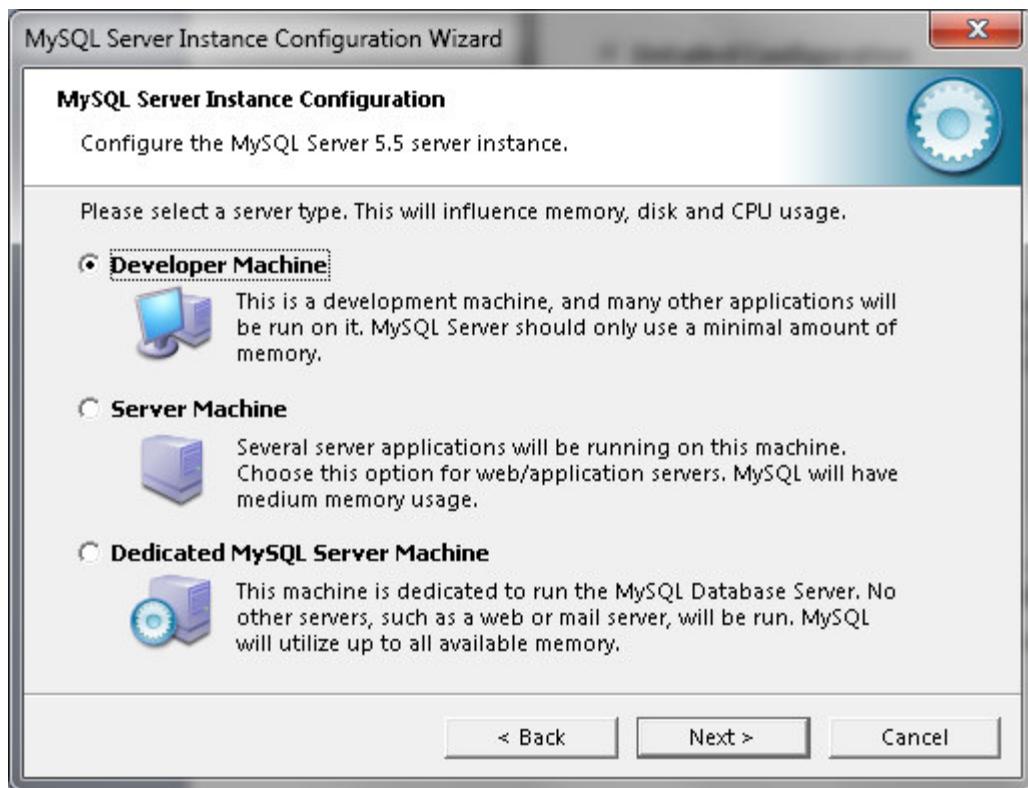


Figura 6.15: Opções de configuração do servidor MySQL.

Passo 4: Será exibida uma tela semelhante à Figura 6.16 exibindo as opções de configuração do banco de dados do servidor MySQL. Selecione a opção de banco de dados multifuncional (*Multifunctional database*), que é a opção mais flexível, e clique no botão *Next* para prosseguir.

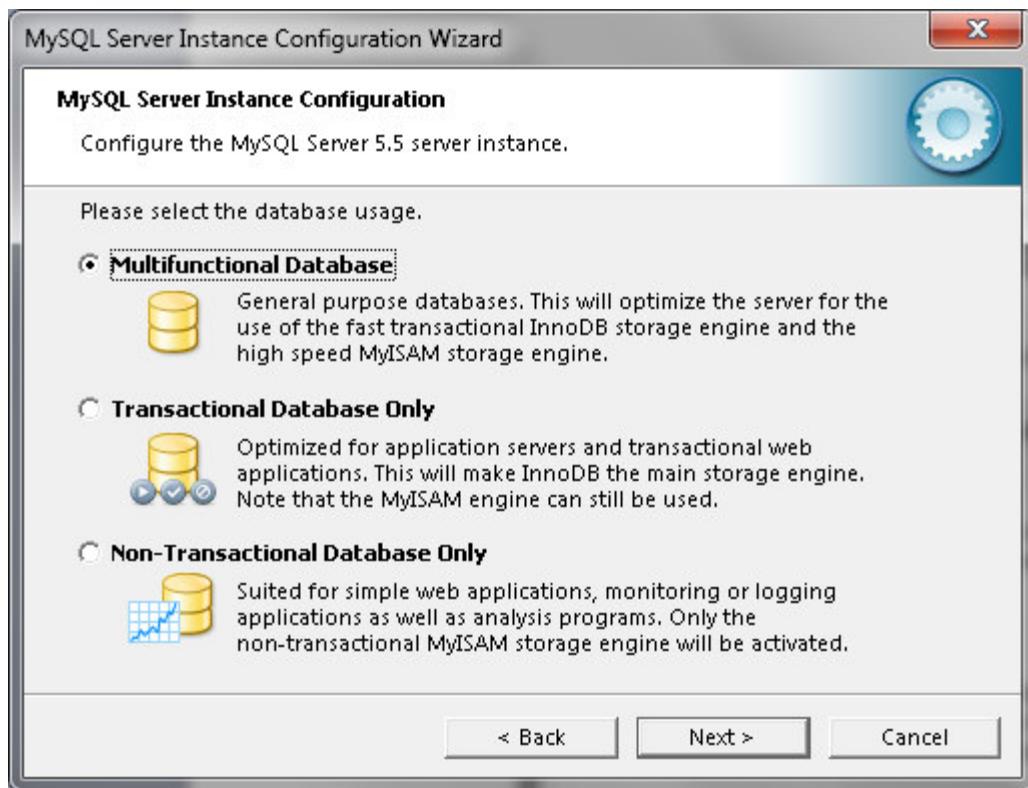


Figura 6.16: Opções de configuração do banco de dados do servidor MySQL.

Passo 5: Surgirá então uma tela como a da Figura 6.17, para configuração do local onde serão armazenados os arquivos correspondentes ao servidor de banco de dados MySQL. A opção padrão armazena esses arquivos dentro da mesma pasta onde foi realizada a instalação da ferramenta MySQL. Mantenha como está e clique no botão *Next* para continuar.

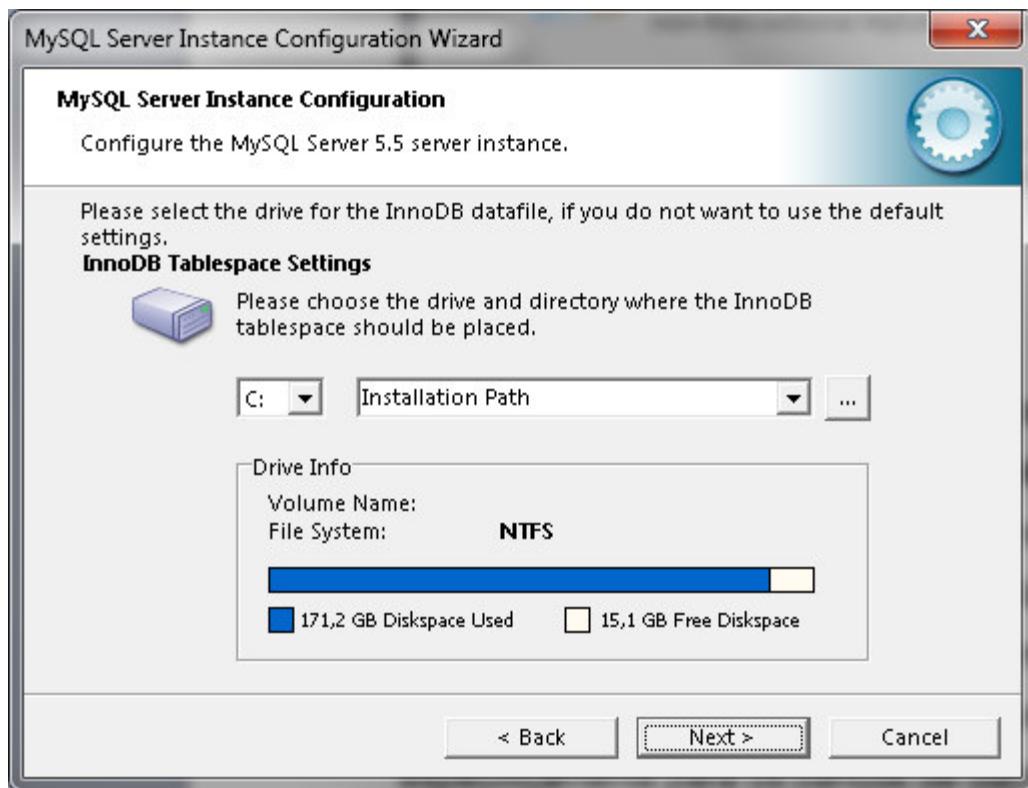


Figura 6.17: Tela de configuração do local dos arquivos de banco de dados do servidor MySQL.

Passo 6: Aparecerá uma nova tela, parecida com a Figura 6.18, exibindo configurações com relação ao número de conexões concorrentes ao servidor de banco de dados. Mantenha selecionada a opção de suporte à decisão (*Decision support*), que consome menos recursos do computador, e clique em *Next* para continuar.

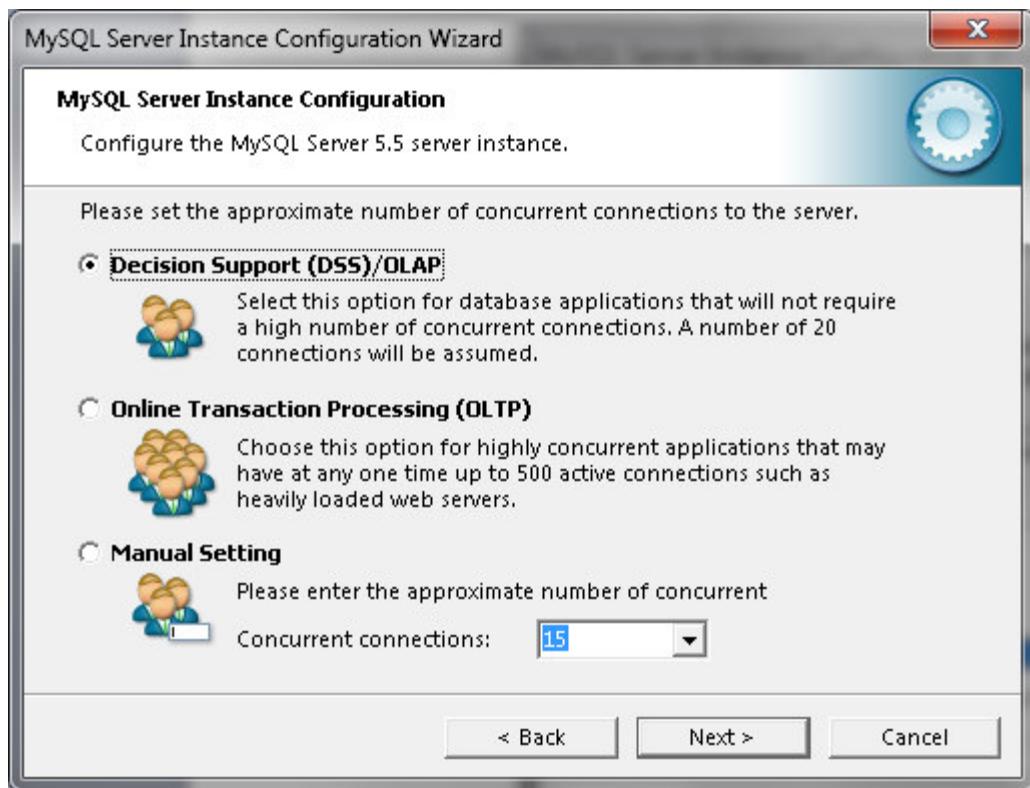


Figura 6.18: Configuração do número de conexões concorrentes permitidas no servidor MySQL.

Passo 7: A tela seguinte (Figura 6.19) está relacionada às possibilidades de acesso ao servidor MySQL pela rede. Nesta tela você pode definir uma porta específica para conexão do servidor MySQL com as demais máquinas que desejam acessá-lo. Mantenha as opções para ativar o servidor na rede TCP/IP (*Enable TCP/IP networking*) e ativar modo restrito (*Enable strict mode*) selecionadas e clique em *Next* para prosseguir.

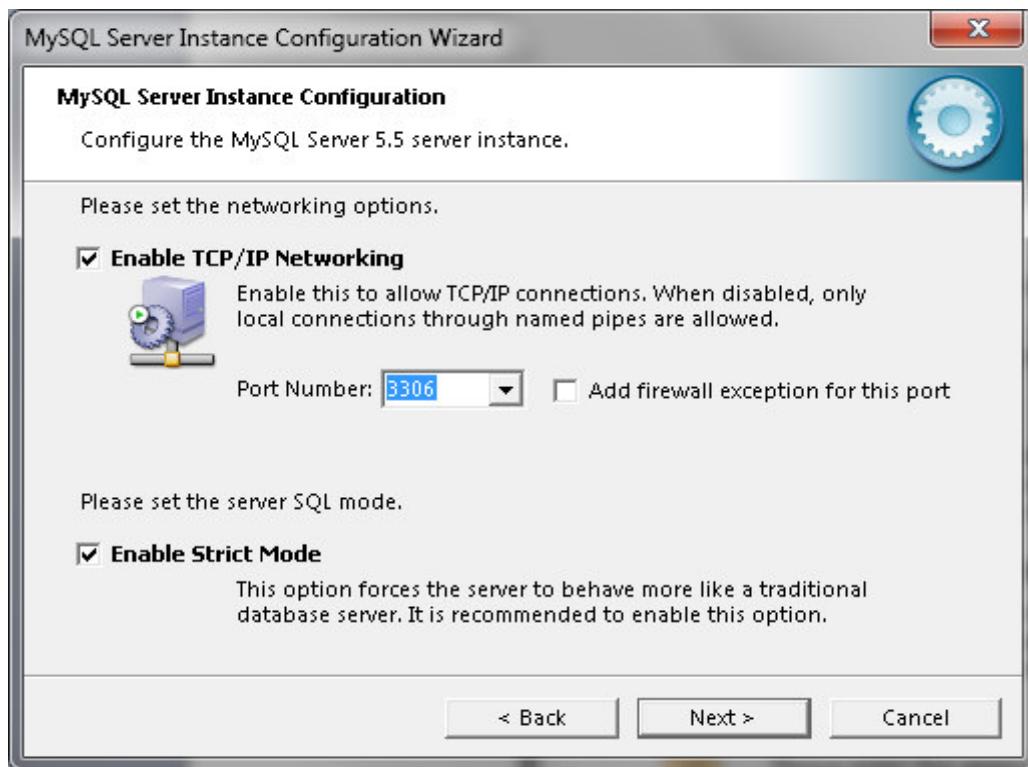


Figura 6.19: Configuração da rede TCP/IP.

Passo 8: Aparecerá uma tela de opções de idioma do servidor (Figura 6.20) de banco de dados. Aqui você pode modificar a opção de idioma desejada ou prosseguir com a configuração do banco de dados. Mantenha selecionada a opção conjunto de caracteres padrão (*Standard character set*) e clique no botão *Next*.

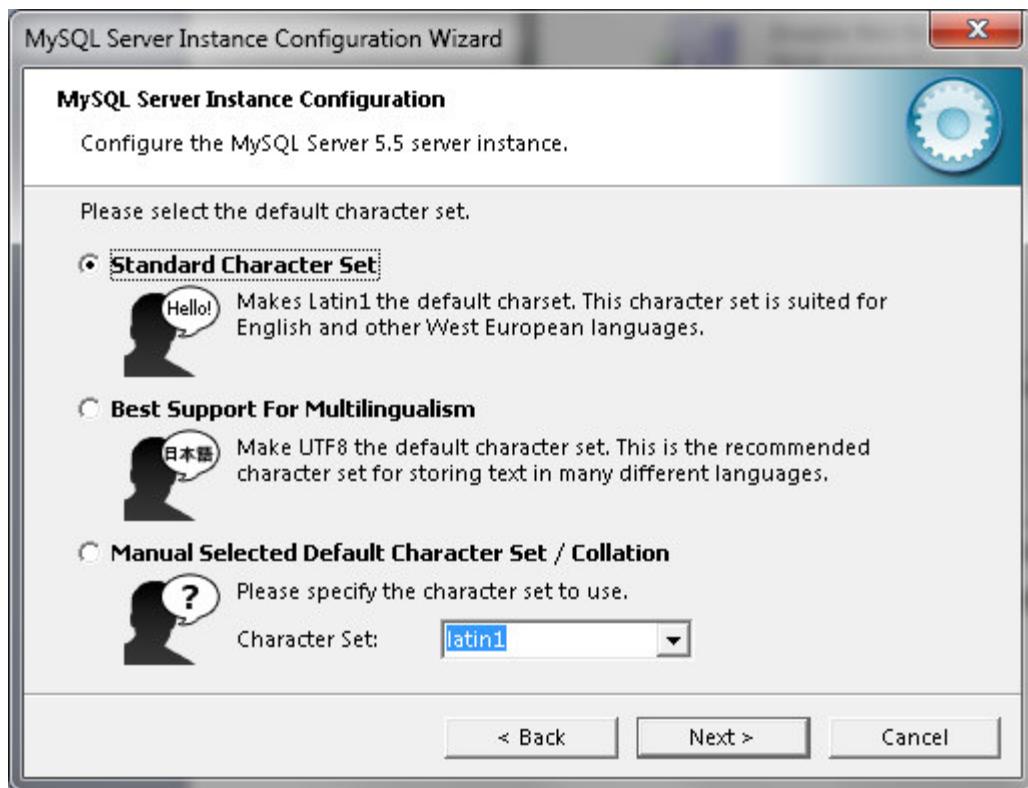


Figura 6.20: Opções de configuração do idioma do banco de dados.

Passo 9: A tela seguinte (Figura 6.21) exibe configurações que permitem que o servidor de banco de dados MySQL seja executado como um serviço no computador ou diretamente pelo *prompt* de comando (terminal). Você também pode escolher um nome para o serviço de execução do servidor MySQL. Deixe marcada a opção *Install as Windows Service* e clique no botão *Next* para prosseguir.

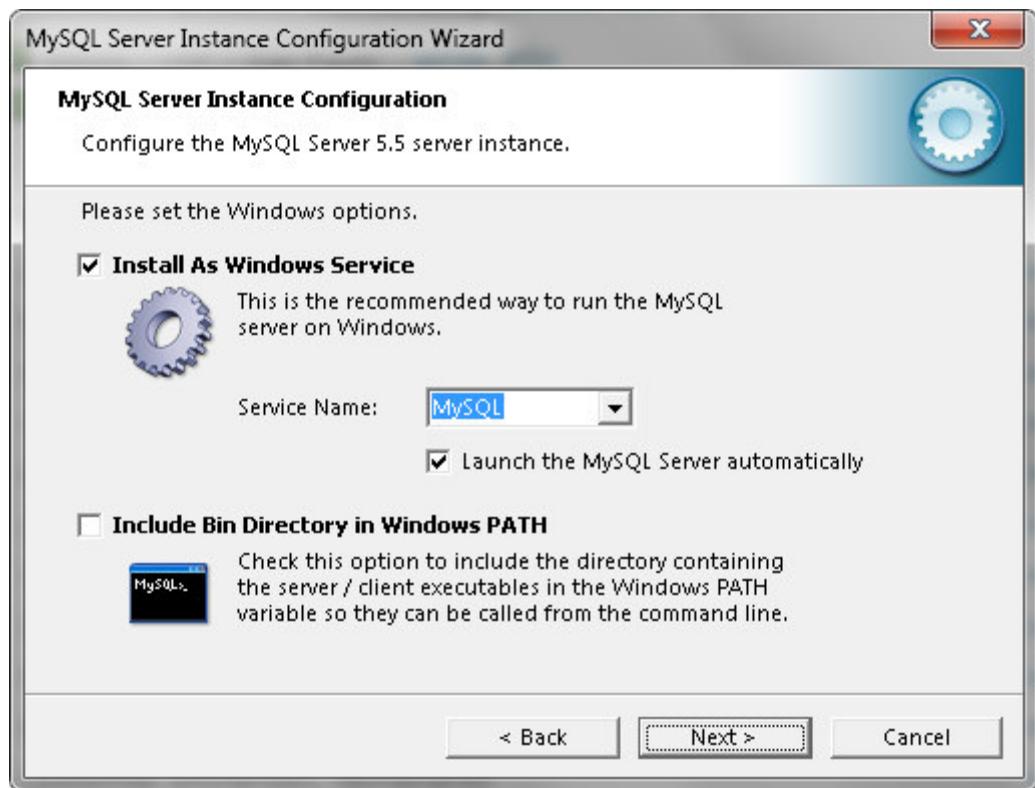


Figura 6.21: Configuração da ferramenta MySQL como um serviço.

Passo 10: Agora aparece a opção de configurar uma senha (Figura 6.22) para o administrador do banco de dados. Para tal, preencha os campos Senha e Confirmação da senha com o valor *admin* e clique em *Next* para seguir adiante.

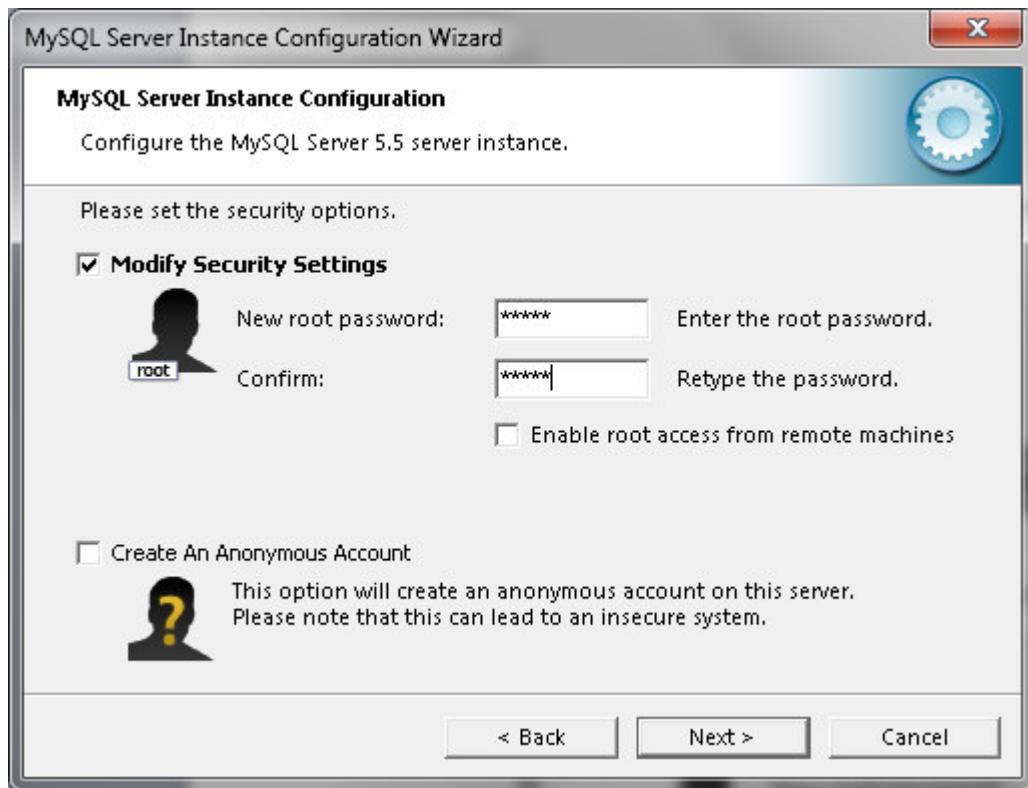


Figura 6.22: Criando uma conta para o administrador do banco de dados.

Passo 11: A última tela é um sumário das configurações selecionadas para o servidor de banco de dados MySQL. Pode ser visualizada na Figura 6.23. Clique no botão *Execute* para iniciar a configuração do servidor de banco de dados com as opções que foram escolhidas por você;

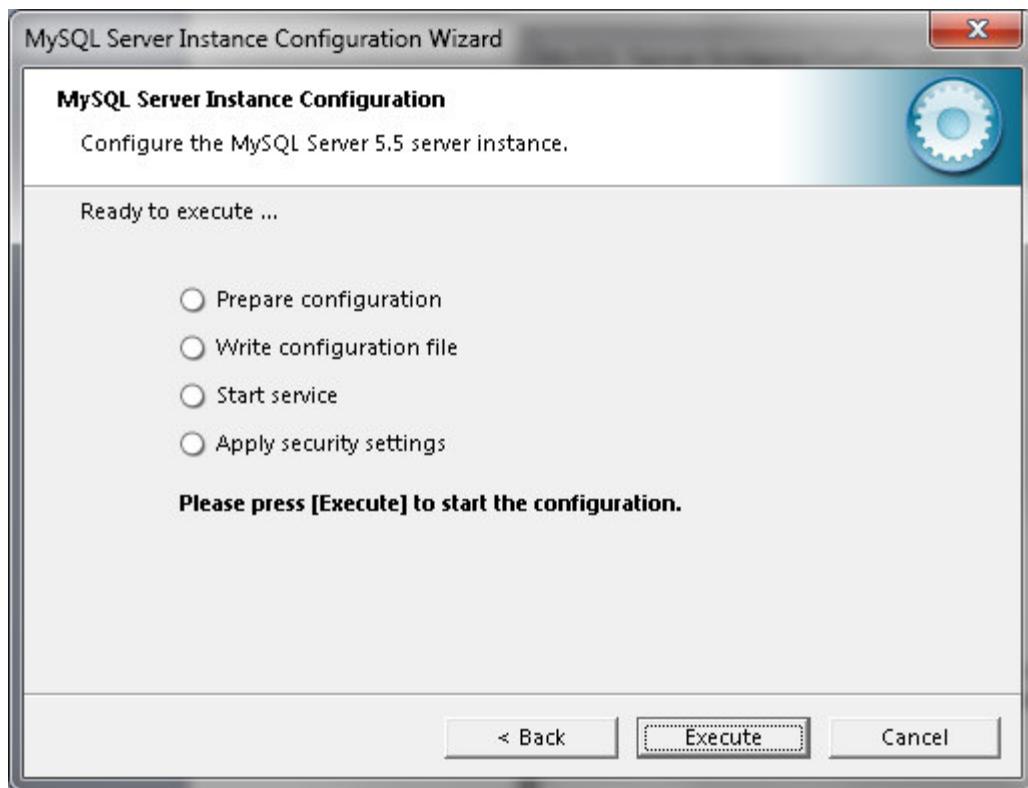


Figura 6.23: Execução das configurações programadas para o servidor MySQL.

Durante a execução, os itens já configurados vão sendo marcados com a cor azul. Ao final da execução, é exibido o botão *Finish*, conforme pode ser visto na Figura 6.24. Clique nesse botão para encerrar o processo de configuração da base de dados MySQL.

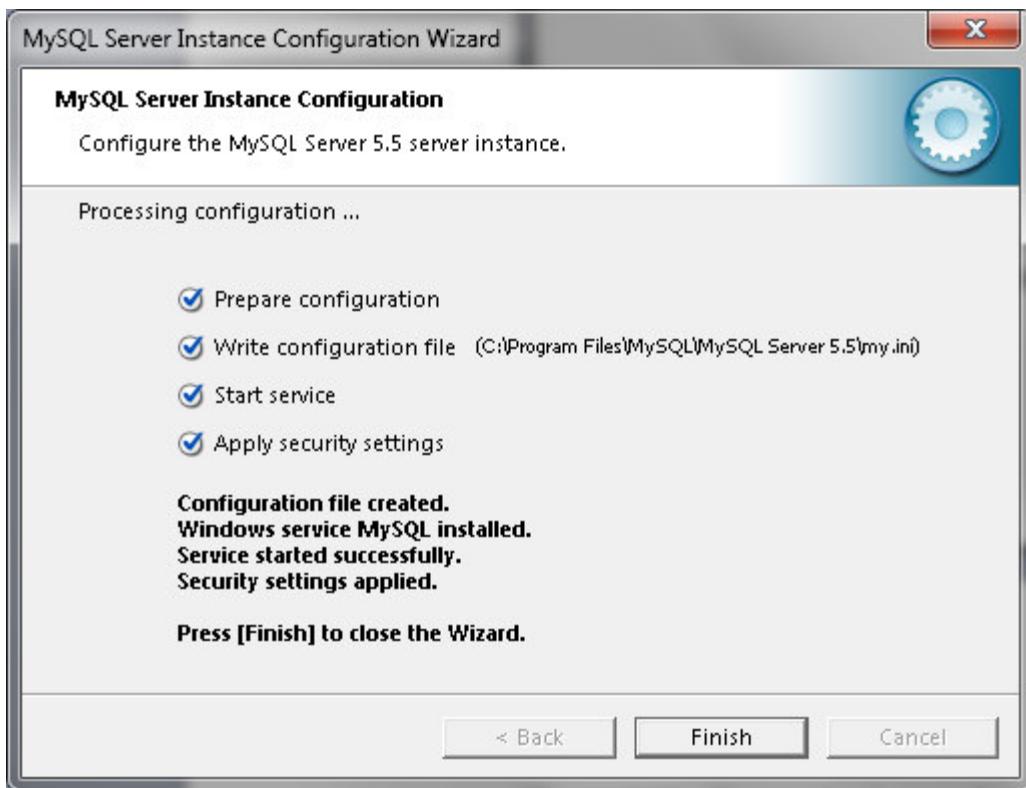


Figura 6.24: Tela indicadora dos serviços de configuração do banco de dados MySQL.

Início da atividade

Atividade Online 2 - Atende ao objetivo 2

Agora que você visualizou os passos para configurar a ferramenta MySQL, faça a configuração da ferramenta no seu computador.

Fim da atividade

Parabéns! Agora a ferramenta MySQL já está instalada e configurada no seu computador! Para acessar o servidor, chame a opção *MySQL Command Line Client*, digite a senha *admin*. Você estará conectado ao servidor de banco de dados MySQL. A Figura 6.25 mostra a janela de linha de comando para o servidor MySQL.

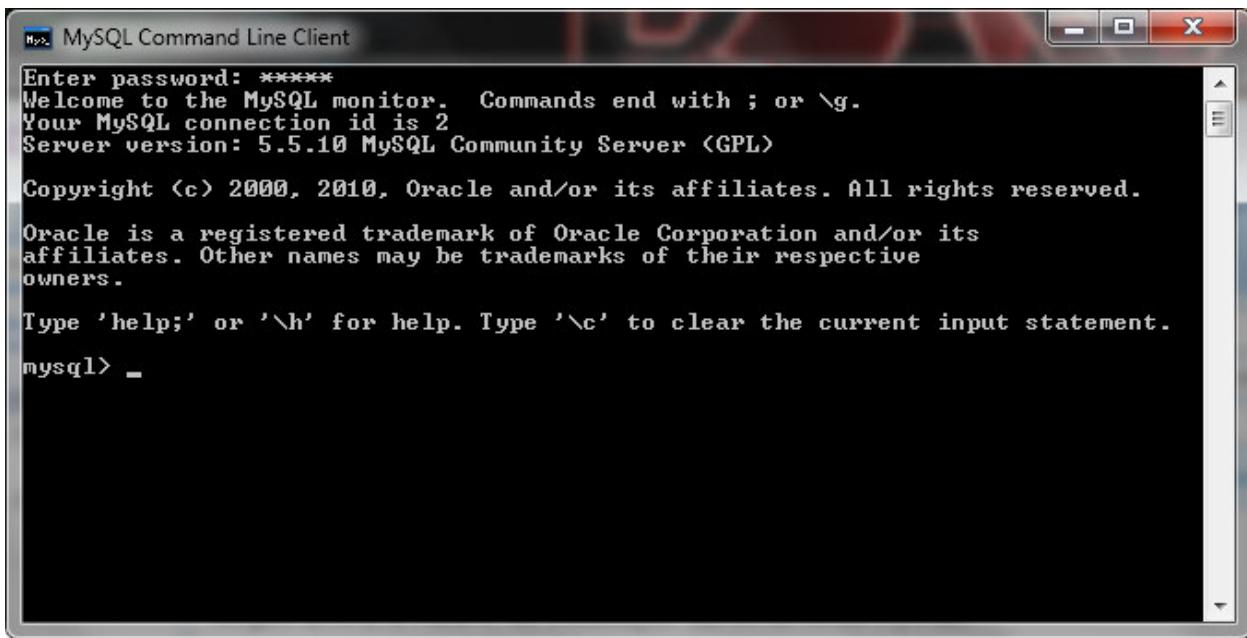


Figura 6.25: Janela de linha de comando para o servidor MySQL.

Conclusão

Nesta aula, você aprendeu a realizar a instalação da ferramenta MySQL e a configurar um servidor de banco de dados. As próximas aulas serão baseadas em comandos SQL (a serem executados dentro do servidor que você criou) para poder realizar o projeto físico do banco de dados.

Resumo

Etapa 1: Baixando o arquivo de Instalação da ferramenta MySQL

Para iniciar a instalação da ferramenta, é necessário antes baixar o arquivo de instalação. Para tal, execute os seguintes passos:

- Acesse o link <http://www.mysql.com/downloads/mysql/> e verifique se o sistema operacional instalado no seu computador é de 32 bits ou 64 bits. Você pode fazer isso acessando a opção Sistema e Segurança / Sistema no painel de controle;
- Em seguida, clique no botão *Download* referente ao arquivo “MSI Installer” correspondente (um dos primeiros da lista). Para pular a etapa de identificação do usuário, clique no link *No thanks, just take me to the downloads!*, que é exibido ao final da página;

- Escolha um dos servidores brasileiros para baixar o arquivo de instalação da ferramenta MySQL e clique no link HTTP correspondente a ele para baixar o arquivo.

Etapa 2: Instalação da ferramenta MySQL

Agora que você já terminou de baixar o arquivo de instalação da ferramenta MySQL, clique duas vezes sobre ele para iniciar o processo de instalação e siga os seguintes passos:

- Será exibida uma janela *pop-up* de boas-vindas; clique no botão *Next* para prosseguir;
- Aparecerá um termo de licença de uso da ferramenta MySQL. Aceite os termos da licença de uso e clique novamente no botão *Next*;
- Aparecerá uma tela para a escolha do modo de instalação a ser realizado no seu computador. Clique no botão *Typical* para selecionar o modo de instalação típico;
- Clique no botão *Install* para iniciar a instalação. Após surgir a nova janela *pop-up* de propaganda do serviço de suporte à ferramenta MySQL, clique no botão *Next* nesta tela e na tela seguinte;
- Ao final da instalação, marque a opção *Launch the MySQL instance configuration wizard* e clique no botão *Finish* para iniciar o processo de configuração da ferramenta.

Etapa 3: Configuração da ferramenta MySQL

Você já baixou e instalou a ferramenta MySQL. Vamos partir para a configuração das opções existentes. Para tal, siga estes passos:

- Após a instalação, será exibida uma nova janela *pop-up* de configuração do servidor MySQL. Clique no botão *Next* para iniciar o processo de configuração da ferramenta;
- Aparecerá agora a tela de opções de configuração do servidor de banco de dados MySQL. A ferramenta MySQL permite que a configuração seja realizada de forma detalhada ou na opção padrão. Mantenha selecionada a opção de

configuração detalhada (*Detailed configuration*) e clique no botão *Next*;

- Selecione o tipo de servidor MySQL - opção *Developer Machine* e clique em *Next* para seguir adiante;
- Selecione a opção de banco de dados multifuncional (*Multifunctional database*), que é a opção mais flexível, e clique no botão *Next* para prosseguir;
- Surgirá uma tela para configuração do local onde serão armazenados os arquivos correspondentes ao servidor de banco de dados MySQL. A escolha padrão armazena esses arquivos dentro da mesma pasta onde foi realizada a instalação da ferramenta MySQL. Mantenha como está e clique no botão *Next* para continuar;
- Mantenha selecionada a opção de suporte à decisão (*Decision support*), que consome menos recursos do computador, e clique em *Next* para continuar;
- Mantenha as opções para ativar o servidor na rede TCP/IP (*Enable TCP/IP networking*) e ativar modo restrito (*Enable strict mode*) selecionadas e clique em *Next* para prosseguir;
- Aparecerá uma tela de opções de idioma do servidor. Mantenha selecionada a opção conjunto de caracteres padrão (*Standard character set*) e clique no botão *Next*;
- Deixe marcada a opção *Install as Windows Service* e clique no botão *Next* para prosseguir;
- Agora deve-se configurar uma senha para o administrador do banco de dados. Para tal, preencha os campos senha e confirmação da senha com o valor *admin* e clique em *Next* para seguir adiante;
- Clique no botão *Execute* para iniciar a configuração do servidor de banco de dados com as opções que foram escolhidas por você.

Informações sobre a próxima aula

Na próxima aula, você verá como executar comandos simples de pesquisa a dados utilizando a ferramenta Visio.

Referências bibliográficas

DATE, C.J. *Introdução a Sistemas de Bancos de Dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.

CARVALHO, C.R. *SQL - guia prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.

ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.

HEUSER, C.A. *Projeto de banco de dados*. 4^a ed. Porto Alegre: Bookman, 2009.

SETZER, V. W.; CORRÊA DA SILVA, F. S. 2005. *Bancos de dados*. São Paulo: Edgard Blucher, 2005.

SILBERSCHATZ, A.; KORTH, H. *Sistema de banco de dados*. 3^a ed. São Paulo: Pearson Makron Books, 2008.

Disciplina de extensão: Modelando e implementando bancos de dados relacionais
Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 7

Mapeamento do modelo relacional para o projeto físico (Parte 1 - Criação de tabelas)

Meta

Apresentar conceitos importantes e os passos principais na construção do projeto físico a partir do modelo relacional.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Descrever os diversos tipos de dados;
2. Aplicar as restrições de integridade ao modelo físico.
3. Aplicar a linguagem de definição de dados DDL;
4. Construir o modelo físico a partir dos modelos conceitual e relacional através dos comandos de criação de tabelas.

Pré-requisitos

Para o melhor aprendizado desta aula, é importante você relembrar conceitos de modelo relacional, tais como: relação ou tabela, coluna ou atributo, linhas ou tuplas, domínio, tipos de chaves, conceito de nulo. Esses conceitos foram estudados na aula 5.

A última etapa: o modelo físico

Conforme você viu nas aulas anteriores, o projeto de banco de dados é composto por três etapas ou fases: conceitual, lógico e físico. As duas primeiras etapas foram descritas em aulas anteriores, buscando oferecer uma visão abstrata de dados aos usuários, ou seja, isolando-os de certos detalhes do banco de dados.

A última etapa refere-se ao nível mais baixo, isto é, o nível físico, criando assim o modelo físico. Neste modelo, fazemos a modelagem física do modelo de banco de dados, descrevendo a forma como os dados estão realmente armazenados no computador e são direcionados para um **SGBD** (sistema de gerenciamento de banco de dados) específico. Então o modelo físico leva em conta as limitações impostas pelo SGBD escolhido e deve ser criado sempre com base nos exemplos de modelagem de dados produzidos na aula de modelo lógico (aula 6).

Início de Verbete

Sistema de gerenciamento de banco de dados (SGBD)

Do inglês DBMS (*data base management system*), é o conjunto de programas de computador (*softwares*) responsáveis pelo gerenciamento de uma base ou banco de dados. Seu principal objetivo é retirar do cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados.

Fim de Verbete

Por outro lado, decisões tomadas durante o projeto físico para melhorar o desempenho podem afetar a estrutura do esquema lógico. Então, cria-se o modelo físico a partir do modelo lógico, da escolha do SGBD e da linguagem SQL. Veja o esquema a seguir:



Figura 7.1: Projeto físico de banco de dados.

[\(redesenhar\)](#)

O foco desta aula é o desenvolvimento das tarefas do nível físico, descrevendo a forma como os dados são armazenados, criando, visualizando, alterando e excluindo tabelas.

Serão descritos aspectos físicos de implementação, utilizando a linguagem padrão SQL.

Fim da Introdução

Diversos tipos de dados

Você já viu que, em bancos de dados relacionais, as informações ficam armazenadas em diversas tabelas. As tabelas têm linhas e colunas e a cada coluna de uma tabela está associado um tipo de dado, fazendo com que somente dados que sejam do tipo adequado possam ser armazenados na coluna.

Então, para cada campo coluna é necessário determinar o tipo de dados que contém, para poder ajustar a estrutura da base de dados e conseguir um armazenamento com a menor utilização de espaço.

O detalhamento de uma tabela deve apresentar todas as colunas juntamente com seu tipo de dado e o tamanho correspondente. 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.

| <i>Padrão</i> | <i>SQL Ansi</i> | <i>DBF</i> | <i>Paradox</i> | <i>MS-Access</i> | <i>Oracle</i> | <i>MS-SQL Server</i> | <i>Interbase</i> | <i>MySQL</i> |
|---------------|-----------------------------|------------|-----------------------------|---------------------|--------------------------|------------------------|-----------------------------|------------------------------|
| <i>Tipo</i> | 92 | III+ | 7 | 2000 | 8 | 7 | 4.2 | 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 |

Figura 7.2: Principais tipos de dados suportados por alguns SGBDs relacionais.

Veja na Figura 7.2 que os diversos tipos de dados (texto, data, hora etc.) podem ser utilizados nos diferentes SGBD relacionais. Por isso o modelo físico depende do SGBD escolhido.

Entenda o significado dos diferentes tipos de dados:

- INTEGER ou INT: número positivo ou negativo inteiro.
- NUMERIC: número positivo ou negativo de **ponto flutuante**. Normalmente, deve-se informar o tamanho total do campo e definir quantas casas decimais devem ser armazenadas após a vírgula.
- DECIMAL: semelhante ao NUMERIC; mas, em alguns bancos de dados, poderá ter maior precisão depois da vírgula.
- CHAR: permite armazenar cadeias de caracteres (letras, combinação de símbolos, letras e números). O tamanho deve ser informado e será fixo, ou seja, mesmo que não utilizado totalmente, o espaço será ocupado fisicamente.
- VARCHAR: permite armazenar cadeias de caracteres, mas com tamanho variável. Se for utilizado menos espaço que o máximo definido, o espaço restante não será ocupado.

Início do Verbete

Número de ponto flutuante

É um formato de representação digital de números reais usado nos computadores.

Fim do Verbete

Exemplo:

| Nome Campo | Tipo | Tamanho | Representação | Descrição |
|----------------|---------|----------------------------------|---------------|--|
| NomeEstudante | VARCHAR | 30 | VARCHAR(30) | Nomes próprios de pessoas. |
| IdadeEstudante | INTEIRO | 2 | INTEIRO | Valor inteiro entre 14 e 60 representando a idade de pessoa em anos. |
| PesoEstudante | DECIMAL | 2 inteiros mais 2 casas decimais | DECIMAL(2,2) | Valor decimal entre 40 e 80 representando o peso de pessoas em quilos. |
| Telefone | CHAR | 10 | CHAR(10) | Número de telefone válido no Brasil. |
| Departamento | CHAR | 4 | CHAR(4) | Siglas dos departamentos existentes num determinado lugar. |

Figura 7.3. Exemplo: Diversos tipos de dados.

Observação:

- O nome de estudante é definido como VARCHAR de tamanho 30, ou seja, se o nome de um estudante tiver menos que 30 caracteres, o espaço restante não será ocupado. Por outro lado, se o nome de estudante fosse definido como CHAR de tamanho 30 e um nome tivesse menos de 30 caracteres, o espaço restante é reservado como se tivesse 30 caracteres. Então, para evitar desperdício de espaço, é conveniente utilizar VARCHAR;
- A idade do estudante é um campo definido como um número inteiro de tamanho ou comprimento 2, pois estamos considerando que as idades podem flutuar entre 14 e 60 anos. Não estamos considerando meses nem dias;
- O peso do estudante não é sempre expressado como um valor inteiro exato. Por isso, deve ser considerado um número com casas decimais;
- O telefone pode ser considerado como dado do tipo CHAR de tamanho 10, pois sabemos que os números telefônicos no Brasil possuem 10 caracteres, incluindo o código da cidade;
- Como é conhecido o tamanho exato do nome de departamento, neste caso deve ser definido como CHAR de tamanho 4.

Por outro lado, os dados SQL se classificam em 13 tipos de dados primários. Veja os diversos tipos de dados, longitude (espaço utilizado para armazenamento do tipo de dado escolhido) e sua descrição:

| Tipo de Dados | Longitude | Descrição |
|---------------|----------------------|--|
| BINARY | 1 byte | Para consultas sobre tabela anexa de produtos de banco de dados que definem um tipo de dados Binário. |
| BIT | 1 byte | Valores Sim/Não ou True/False |
| BYTE | 1 byte | Um valor inteiro entre 0 e 255. |
| COUNTER | 4 bytes | Um número incrementado automaticamente (de tipo Long) |
| CURRENCY | 8 bytes | Um inteiro escalável entre 922.337.203.685.477,5808 e 922.337.203.685.477,5807. |
| DATETIME | 8 bytes | Um valor de data ou hora entre os anos 100 e 9999. |
| SINGLE | 4 bytes | Um valor em ponto flutuante de precisão simples com uma classificação de - 3.402823*1038 a -1.401298*10-45 para valores negativos, 1.401298*10- 45 a 3.402823*1038 para valores positivos, e 0. |
| DOUBLE | 8 bytes | Um valor em ponto flutuante de dupla precisão com uma classificação de - 1.79769313486232*10308 a -4.94065645841247*10-324 para valores negativos, 4.94065645841247*10-324 a 1.79769313486232*10308 para valores positivos, e 0. |
| SHORT | 2 bytes | Um inteiro curto entre -32.768 e 32.767. |
| LONG | 4 bytes | Um inteiro longo entre -2.147.483.648 e 2.147.483.647. |
| LONGTEXT | 1 byte por caractere | De zero a um máximo de 1.2 gigabytes. |
| LONGBINARY | Segundo se necessite | De zero 1 gigabyte. Utilizado para objetos OLE. |
| TEXT | 1 byte por caractere | De zero a 255 caracteres. |

Figura 7.4: Treze tipos de dados primários - SQL.

Outros tipos de dados:

- FLOAT: número de ponto flutuante de simples precisão. Não confundir com o DOUBLE, que também é utilizado para números de ponto flutuante, mas de dupla precisão, ou seja, maior aproximação de resultados.
- DATE: permite armazenar datas.
- TIME: permite armazenar horários.

Exemplo: Considere a tabela a seguir e realize a descrição dos tipos de dados utilizados em cada coluna.

| CURSO | Nome | Número | Créditos | Departamento |
|-------|--------------------|----------|----------|--------------|
| | Algoritmo | DCC2341 | 4 | DCC |
| | Estrutura de Dados | DCC 5613 | 4 | DCC |
| | Matemática | MAT2131 | 4 | MAT |
| | Base de Dados | DCC 6765 | 4 | DCC |

Figura 7.5: Tabela da disciplina.

Observe que a tabela CURSO apresenta quatro campos.

- O primeiro campo é o nome do curso. Pode ser definido como VARCHAR de tamanho 30 ==> VARCHAR(30);
- O segundo campo é o número ou código de curso, formado de 3 letras e 3 números. Pode ser definido como CHAR (*contém letras e números*) de tamanho 7 ==> CHAR(7).
- O terceiro campo é a informação de créditos dos cursos. Observe que ele contém um só dígito e pode ser definido como INTEIRO de tamanho 1 ==> INTEIRO.
- O quarto campo é o departamento ao qual pertence cada curso, formado

exatamente por três letras. Então pode ser definido como CHAR de tamanho 3
==> CHAR(3).

Finalmente obtemos as seguintes descrições dos tipos de dados de cada campo:

| Campo | Tipo de dado | Tamanho | Representação |
|--------------|--------------|---------|---------------|
| Nome | VARCHAR | 30 | VARCHAR(30) |
| Número | CHAR | 7 | CHAR(7) |
| Créditos | INTEIRO | 1 | INTEIRO |
| Departamento | CHAR | 3 | CHAR(3) |

Exemplo: Considere o seguinte diagrama ER e seus respectivos esquemas relacionais.



Esquemas relacionais: PROFESSOR (Número, Nome, Endereço)

CURSO (Ident, Nome)

Figura 7.6: Diagrama ER e esquema relacional.

Não está explícito o conteúdo das tabelas PROFESSOR e CURSO, mas, em termos gerais, podemos definir os seguintes tipos de dados:

- Na tabela PROFESSOR, o primeiro campo é a matrícula do professor. Suponha que esse campo seja formado por números de comprimento ou tamanho 6. Pode-se considerar como tipo CHAR, pois em geral aquele número serve para identificação do professor e não é utilizado para realizar alguma conta aritmética ==> CHAR(6);
- Na tabela PROFESSOR, o segundo campo é o nome do professor ==> VARCHAR(35);
- Na tabela PROFESSOR, o terceiro campo é o endereço ==> VARCHAR(40);

- Na tabela CURSO, o primeiro campo é a identidade ou código do curso. Suponha que esse campo seja formado pelas três primeiras letras do departamento e os três números restantes referem-se à ordem sequencial de criação dele ==> CHAR(6).
- Na tabela CURSO, o segundo campo é o nome do curso ==> VARCHAR(25).

Finalmente, obtemos as seguintes descrições dos tipos de dados para cada tabela:

PROFESSOR

| Nome Campo | Tipo de dado | Tamanho | Representação |
|------------|--------------|---------|---------------|
| Número | CHAR | 6 | CHAR(6) |
| Nome | VARCHAR | 35 | VARCHAR(35) |

CURSO

| Nome Campo | Tipo de dado | Tamanho | Representação |
|------------|--------------|---------|---------------|
| Identidade | CHAR | 6 | CHAR(6) |
| Nome | VARCHAR | 25 | VARCHAR(25) |

Vamos colocar em prática os conceitos estudados até aqui.

Inicio da atividade

Atividade 1 - Atende ao objetivo 1

Considere a seguinte tabela e realize as descrições dos tipos de dados.

| Vôo | Tarifa | Assento | Cia | Nome | Ano Fabricação | Presid | Sede |
|-------|---------|---------|-----|------------|----------------|--------|--------------|
| AF147 | 1000,00 | 250 | 128 | Air France | 1960 | Etoile | Paris |
| AF455 | 750,00 | 500 | 128 | Air France | 1985 | Etoile | Paris |
| RG224 | 500,00 | 150 | 423 | Varig | 1989 | Pampa | Porto Alegre |

Diagramação: favor deixar 10 linhas para a resposta.

Resposta comentada

Observe que a tabela apresenta oito campos.

- O primeiro campo é o número do voo, que contém letras e números (exatamente duas letras e três números) ==> CHAR(5);
- O segundo campo é o preço da tarifa, mostrando quatro inteiros e dois decimais ==> DECIMAL(4,2);
- O terceiro campo é a quantidade de assento disponíveis ==> INTEGER.

- O quarto campo é o código da companhia ==> CHAR(3).
- O quinto campo é o nome da companhia ==> VARCHAR(10).
- O sexto campo é o ano de fabricação do aparelho ==> INTEGER.
- O sétimo campo é o nome do presidente ==> VARCHAR(10).
- O oitavo campo é a sede ==> VARCHAR(12).

Finalmente, obtemos as seguintes descrições dos tipos de dados correspondentes:

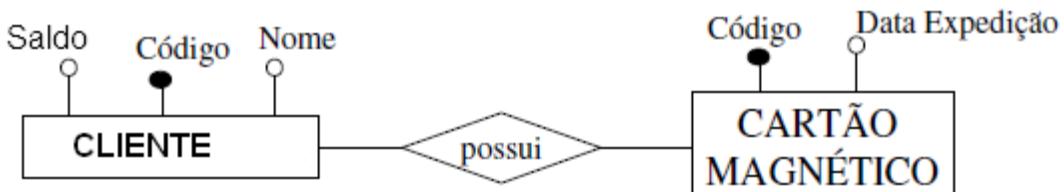
| Nome Campo | Tipo de dado | Tamanho | Representação |
|----------------|--------------|----------------------|---------------|
| Voo | CHAR | 5 | CHAR(5) |
| Tarifa | DECIMAL | 4 e 2 casas decimais | DECIMAL(4,2) |
| Assento | INTEIRO | 3 | INTEIRO |
| Cia | CHAR | 3 | CHAR(3) |
| Nome | VARCHAR | 10 | VARCHAR(10) |
| Ano fabricação | INTEIRO | 4 | INTEGER |
| Presid | VARCHAR | 10 | VARCHAR(10) |
| Sede | VARCHAR | 12 | VARCHAR(12) |

Fim da atividade

Início da atividade

Atividade 2 - Atende ao objetivo 1

Considere o seguinte diagrama ER e seus respectivos esquemas relacionais. Faça as descrições dos tipos de dados utilizados.



Esquemas relacionais: CLIENTE (Código, Nome, Saldo)

CARTÃO MAGNÉTICO (Código, Data expedição)

Diagramação: favor deixar 10 linhas para a resposta.

Resposta comentada

Observamos, pelo diagrama ER e seus esquemas relacionais:

- Na tabela *CLIENTE*, o primeiro campo, *Nome* ==> *VARCHAR(30)*;
- Na tabela *CLIENTE*, o segundo campo, *Código*, provavelmente 6 dígitos ==> *INTEGER*;
- Na tabela *CLIENTE*, o terceiro campo, *Saldo* ==> *DECIMAL(4,2)*;
- Na tabela *CARTÃO MAGNÉTICO*, o primeiro campo, *Código*, pode variar de 4 a 8 dígitos ==> *INTEGER*;
- Na tabela *CARTÃO MAGNÉTICO*, o segundo campo, *Data expedição* ==> *DATE*

Finalmente, obtemos as seguintes descrições dos tipos de dados de cada tabela:

CLIENTE

| Nome Campo | Tipo de dado | Tamanho |
|------------|--------------|----------------|
| Nome | VARCHAR | 30 |
| Código | INTEGER | 6 |
| Saldo | DECIMAL | 4 inteiros e 2 |

CARTÃO MAGNÉTICO

Fim da atividade

Restrições de integridade no modelo físico

Dizer que os dados de um banco de dados estão íntegros significa dizer que eles refletem corretamente a realidade representada pelo banco de dados e que são consistentes entre si. Para tentar garantir a integridade de um banco de dados, os SGBD oferecem o mecanismo de restrições de integridade.

Tipos de restrições de integridade

- Integridade de domínio: o valor de um atributo deve pertencer ao domínio;
- Integridade de vazio: para cada atributo deve-se definir se ele pode assumir o valor nulo. Isso implica que o atributo deve ser obrigatório (sem nulos) ou opcional (admitindo nulos);

- Integridade de chave: refere-se à definição das chaves primárias e alternativas, indicando que os atributos ou o conjunto de atributos devem ter valores únicos;
 - No caso da chave primária, a integridade é a unicidade;
 - A definição da chave primária define a restrição de integridade de chave.
- Integridade referencial: indica que os valores dos atributos numa chave estrangeira devem aparecer na chave primária de uma tabela referenciada.
 - Na inclusão e alteração de uma tupla (ou linha) na tabela que contém a chave estrangeira, deve-se garantir que o valor da chave estrangeira seja uma chave primária na tabela referenciada;
 - Na alteração e exclusão de uma tupla na tabela referenciada, deve-se garantir que o valor da chave primária nessa tupla não apareça como chave estrangeira.

Exemplo:

Vamos realizar a análise de restrições de integridade. Considere a seguinte tabela:

EMPREGADO

| Cod_Emp | Nome | Cod_Tar |
|---------|---------|---------|
| 120 | Jailson | 77 |
| 343 | Cleber | 42 |
| 459 | Luis | 77 |
| 530 | Marcela | 77 |

TAREFA

| Cod_Tar | Descrição |
|---------|------------|
| 42 | Secretário |
| 12 | Office-boy |
| 77 | Contador |

Figura 7.7: Tabelas de empregado e tarefas

(redesenhar favor não trocar o conteúdo nem os títulos da coluna)

- Cod_Emp. da tabela EMPREGADO e Cod_Tar. da tabela TAREFA são chaves primárias;
- Cod_Tar. da tabela EMPREGADO é uma chave estrangeira;
- As chaves primárias são atributos obrigatórios. Pela regra de integridade – Integridade de Vazio, todas as instâncias de EMPREGADO e de TAREFA devem estar com esta coluna preenchida. Nesse caso, a chave estrangeira é um atributo obrigatório, porque a regra de existência do relacionamento diz que “cada empregado deve possuir uma tarefa”. Poderia ser opcional, caso o

relacionamento fosse “cada empregado pode possuir uma tarefa”;

- A remoção da instância “Secretário”, da tabela TAREFA, ou a troca do conteúdo de Cod_Tar. da tabela TAREFA, de 42 para 29, por exemplo, causa problemas de integridade referencial (pertence à primeira regra);
- A remoção da instância “Office-boy”, da tabela TAREFA não fere nenhuma regra de integridade, porque não existe nenhuma instância em EMPREGADO com chave igual a 12;
- Em relação à existência do relacionamento do lado da tabela TAREFA, o fato de ser opcional (“cada tarefa PODE ser exercida por vários empregados”), faz com que haja instâncias (“Office-boy”) sem necessariamente estarem associadas a algum empregado.

Início da atividade

Atividade 3 - Atende ao objetivo 2

Analise as restrições do seguinte modelo relacional e responda às seguintes questões:

| Departamento | | | | Local_Departamento | | Projeto | | | |
|--------------|-------|---------|----------|--------------------|---------|----------|-------|---------|---------|
| NomDep | NumID | Gerente | DataInic | NumID | LocalID | NomeProj | NumID | NumDept | LocalID |
| Financ | 11 | Lucio | 12/12/04 | 21 | local11 | Sabat | 11 | 31 | local11 |

Figura 7.8: Tabelas Departamento, Local_Departamento e Projeto

(redesenhar favor não trocar o conteúdo nem os títulos da coluna)

- 1) As chaves primárias das três tabelas podem ser atributos opcionais ou obrigatórios?
- 2) Remover uma instância da tabela Local_Departamento pode causar problemas de integridade referencial?
- 3) Mencione uma violação de restrição de chave estrangeira.

Diagramação, inserir 5 linhas para a resposta.

Resposta comentada

- 1) Observamos que nas três tabelas a chave primária é o NumID (número de identificação). As chaves primárias são atributos obrigatórios. Pela regra de integridade – Integridade de Vazio, todas as instâncias das tabelas Departamento,

Local_Departamento e Projeto devem estar com esta coluna preenchida. Nesse caso, as chaves estrangeiras também são atributos obrigatórios, e, pela regra de existência do relacionamento, “cada departamento deve possuir um local e um nome de projeto”.

2) Pode causar problemas de integridade referencial se a instância a ser removida da tabela Local_Departamento se encontra instanciada também na Tabela de Projeto ou na tabela de Departamento.

3) Na inclusão e alteração de uma linha na tabela que contém a chave estrangeira, deve ser garantido que o valor da chave estrangeira seja uma chave primária na tabela referenciada, ou seja, os atributos da chave estrangeira possuem os mesmos domínios dos atributos da chave primária da outra tabela. Aqui dizemos que os atributos da chave estrangeira referenciam à chave primária da outra tabela referenciada. Em nosso caso, uma violação da regra pode acontecer quando os atributos da chave primária da tabela Local_Departamento não possuírem os mesmos domínios dos atributos da chave estrangeira da tabela Projeto.

Fim da atividade

Considere agora os seguintes esquemas relacionais:

EMPRESA = {EMPREGADO, PROJETO, DEPARTAMENTO, TRABALHA_EM}

EMPREGADO = {CPF, NOMEMP, SALARIO, IDSUPERV, NUMDEPTO}

PROJETO = (IDNUMPJ, NOMEPROJ, LOCALPROJ, IDPROJDEPTO, DATAINICIO)

DEPARTAMENTO = (IDDEPTO, NOMEDEPTO, LOCALDEPTO, IDGERENTE)

TRABALHA_EM = {EMPCPF, PROJNUM, QTEHORAS}

As restrições de integridade referencial surgem dos relacionamentos entre as entidades representadas em esquema de relação. Podemos exibir as seguintes restrições de integridade referencial:

1. O atributo NUMDEPTO da relação EMPREGADO fornece o número do departamento para o qual um empregado trabalha. Portanto, indicamos NUMDEPTO como chave estrangeira da relação EMPREGADO, fazendo corresponder ao valor de

IDDEPTO de alguma tupla na relação DEPARTAMENTO. Isso significa que um valor NUMDEPTO da relação EMPREGADO deve corresponder a um valor da chave primária IDDEPTO da relação DEPARTAMENTO.

EMPREGADO[NUMDEPTO] ---> DEPARTAMENTO[IDDEPTO]

2. Na relação PROJETO, o atributo IDPROJDEPTO refere-se ao número do DEPARTAMENTO que possui um projeto, fazendo corresponder ao valor IDDEPTO da relação DEPARTAMENTO. Então, o atributo IDPROJDEPTO é uma chave estrangeira e o atributo IDDEPTO é uma chave primária.

PROJETO[IDPROJDEPTO] ---> DEPARTAMENTO[IDDEPTO]

3) O atributo PROJNUM da relação TRABALHA_EM fornece o número do projeto para o qual um empregado trabalha. Portanto, PROJNUM é a chave estrangeira da relação TRABALHA_EM, fazendo corresponder ao valor de IDNUMPJ de alguma tupla na relação PROJETO. Isso significa que um valor PROJNUM da relação TRABALHA_EM deve corresponder a um valor da chave primária IDNUMPJ da relação PROJETO.

TRABALHA_EM[PROJNUM] ---> PROJETO[IDNUMPJ]

Em resumo:

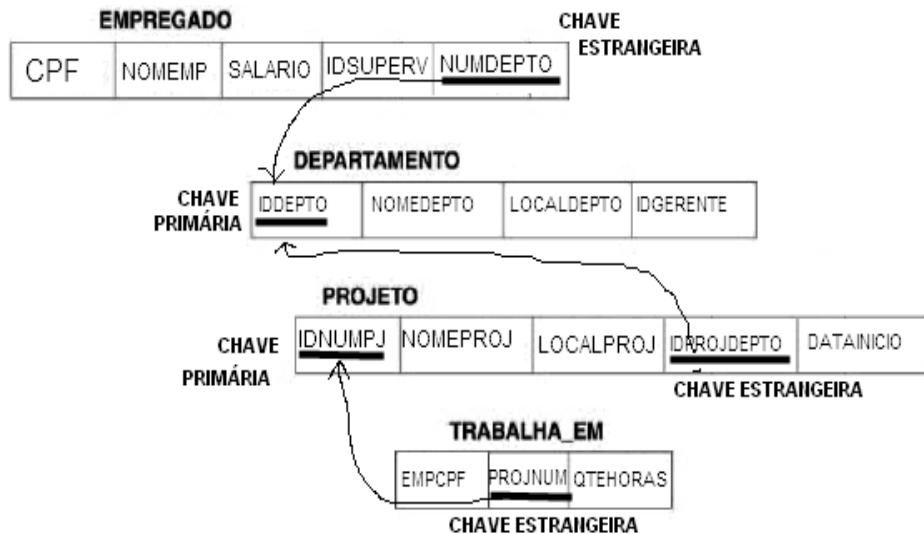


Figura 7.9: Restrições de integridade referencial exibidas no esquema de um banco de dados relacional EMPRESA.

Início da atividade

Atividade 4 - Atende ao objetivo 2

A seguir está um esquema parcial para um banco de dados relacional. Identifique duas restrições de integridade referencial com suas respectivas chaves primárias e chaves estrangeiras:

ALUNO(CodAluno, Nome, CodCurso)

CURSO(CodCurso, Nome)

DISCIPLINA(CodDisciplina, Nome, Creditos, CodDepartamento)

CONCEITO(CodAluno, CodDisciplina, Semestre, Conceito)

DEPARTAMENTO(CodDepartamento, Nome)

Diagramação, inserir 5 linhas para a resposta.

Resposta comentada

Podemos exibir as seguintes restrições de integridade referencial:

1. O atributo *CodAluno* da relação ALUNO fornece a identificação do aluno num certo departamento; indicamos *CodAluno* como chave estrangeira da relação ALUNO, fazendo-o corresponder a um valor da chave primária *CodAluno* de alguma tupla na

relação CONCEITO.

$ALUNO[CodAluno] \rightarrow CONCEITO[CodAluno]$

2. O atributo *CodDepartamento* da relação DISCIPLINA fornece um código ao qual a disciplina pertence num certo departamento; indicamos este atributo como chave estrangeira da relação DISCIPLINA, fazendo-o corresponder a um valor da chave primária *CodDepartamento* de alguma tupla na relação DEPARTAMENTO.

$DISCIPLINA[CodDepartamento] \rightarrow DEPARTAMENTO[CodDepartamento]$

Fim da atividade

Linguagens

A SQL (*Structured Query Language*) é chamada linguagem de consulta por causa do termo em inglês “query”, que poderia ser traduzido para o português como “consulta”. No entanto, esse termo significa mais que recuperar ou consultar dados, mas também atualizá-los. A SQL é formada pelas linguagens: DDL e DML.

Linguagem de Definição de Dados (DDL)

DDL (*Data Definition Language*) – linguagem utilizada para definir a estrutura de armazenamento dos dados, também chamada de dicionário de dados. Através desta linguagem é possível:

- A criação de uma tabela;
- A visualização de uma tabela;
- A alteração do conteúdo de uma tabela;
- A exclusão de uma tabela depois de ser criada.

Criando uma tabela

Ao criarmos uma tabela, devemos especificar:

- seu nome;
- quais as colunas que a compõem;
- quais os tipos de dados das colunas;
- se uma coluna refere-se à chave primária ou não;

- as regras de integridade, entre outros.

Uma coluna definida como de tipo INTEGER pode suportar números inteiros ou o valor NULL. Da mesma forma, uma coluna CHAR suporta cadeias de caracteres e o valor NULL. Lembre-se que NULL indica ausência de valor definido; é diferente de zero. As colunas que não podem conter o valor NULL têm preenchimento obrigatório.

A sintaxe básica do comando para a criação de uma tabela é (a cor azul são os comandos SQL):

```
CREATE TABLE nome_tabela (
    coluna 1 tipo_coluna1 [NOT NULL],
    coluna 2 tipo_coluna2 [NOT NULL],
    coluna 3 tipo_coluna3 [NOT NULL],
);
```

Onde:

- ✓ CREATE TABLE: indica o comando para a criação de uma tabela. A abertura é indicada pelo parênteses "("; o término do comando é indicado pelo fechamento do parênteses ") ";
- ✓ nome_tabela: nome dado à tabela que está sendo criada;
- ✓ coluna 1, coluna 2, coluna 3: nomes das colunas;
- ✓ *tipo_coluna1*, *tipo_coluna2*, *tipo_coluna3*: tipos de dados das colunas coluna 1, coluna 2 e coluna 3;
- ✓ NOT NULL: parâmetro opcional que pode aparecer ao lado do nome e tipo de dados da coluna no momento da criação da tabela. Indica que o valor NULL não é permitido na coluna.

Exemplo:

Considere a tabela a seguir:

| Código do produto | Nome do produto | Preço unitário |
|-------------------|-----------------|----------------|
| 1011 | xnome1 | 2,20 |
| 1012 | xnome2 | 4,50 |

| | | |
|------|--------|------|
| 1009 | xnome3 | 3,10 |
| 1013 | xnome4 | 1,80 |

Figura 7.10: Tabela de produtos

Aplicando o comando de criação com a linguagem padrão SQL, obtemos:

```
CREATE TABLE Produtos (
    codigo_produto INTEGER NOT NULL,
    nome_produto VARCHAR(15),
    preco_unitario DECIMAL(2,2),
);
```

Observa-se o seguinte:

- o nome da tabela é Produtos;
- o primeiro campo tem o nome codigo_produto do tipo inteiro. O valor NULL não é permitido neste primeiro campo, ou seja, o preenchimento é obrigatório;
- o segundo campo é o nome do produto denominado nome_produto do tipo VARCHAR de tamanho 15;
- o terceiro campo é o preço unitário do tipo decimal com dois inteiros e duas casas decimais.

Início da atividade online

Atividade 5 – Atende aos objetivos 2, 3 e 4 (Diagramador, favor inserir ícone de atividade online)

Entre no ambiente virtual e resolva a atividade a seguir enviando sua resposta ao tutor. A partir do minimundo e dos modelos conceitual e lógico apresentados a seguir, descreva os comandos necessários para a criação do modelo físico correspondente.

Minimundo 03

A empresa AutoRun é especializada no conserto de veículos automotores, incluindo carros, caminhões e motos. Devido a recentes prejuízos com o desaparecimento de peças na empresa, o gerente decidiu implantar um sistema para controlar os orçamentos emitidos e as peças utilizadas nos respectivos serviços prestados pela

empresa.

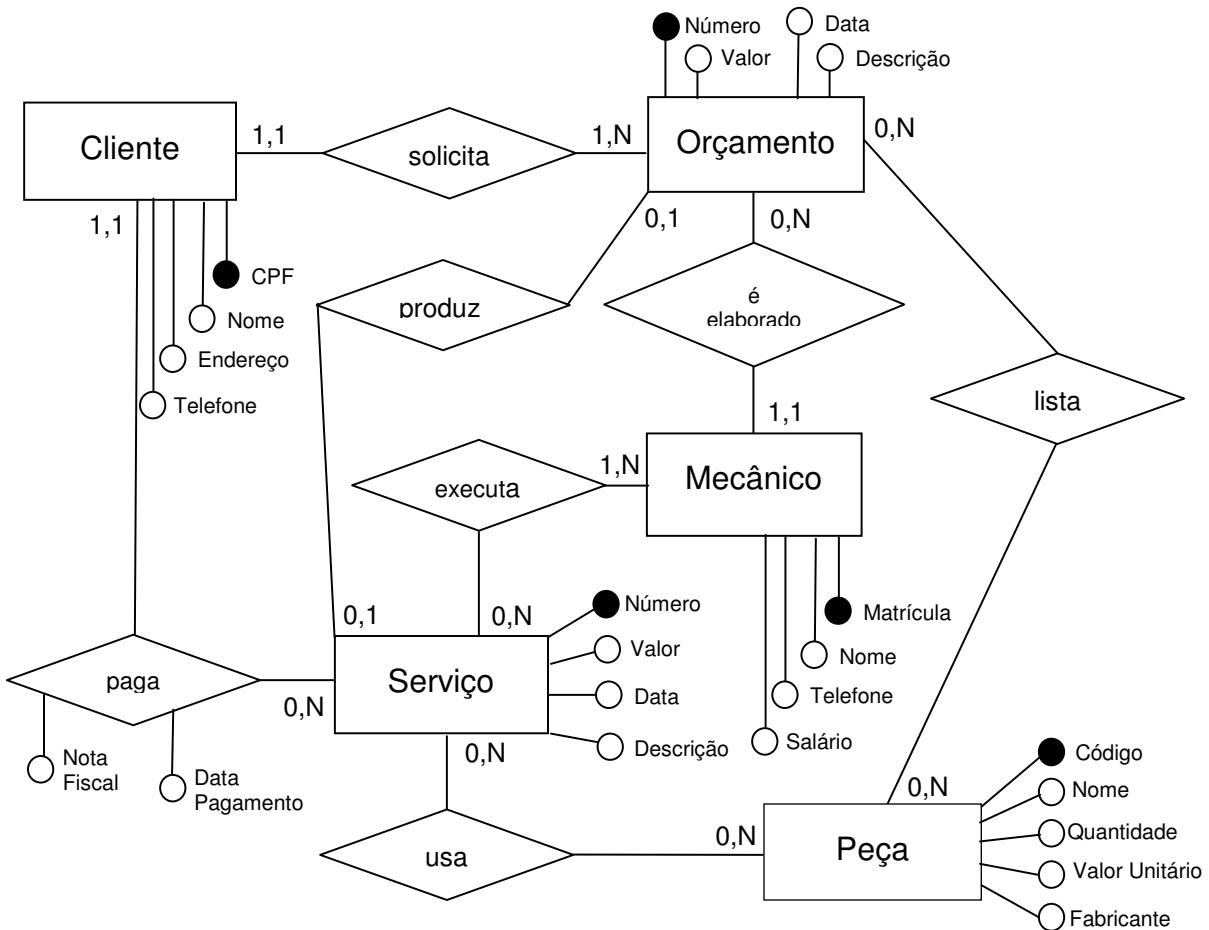
O gerente informou que, tanto no caso dos orçamentos quanto para os serviços realizados, normalmente o mecânico responsável toma nota do número do orçamento ou serviço, o valor total correspondente e a data da sua realização e acrescenta uma descrição detalhada do que foi orçado ou realizado. Ele também precisa relacionar todas as peças que foram previstas no orçamento ou utilizadas na execução do serviço, se for o caso.

No caso da solicitação de um orçamento, as seguintes informações do cliente também são registradas pelo mecânico: CPF, nome do cliente, endereço e telefone para contato. No caso de um serviço, também são anotados o número do orçamento correspondente (se houver) e os nomes de todos os mecânicos que o executaram. Ao final do serviço, é emitida uma nota fiscal que deverá ser paga pelo cliente.

O gerente informou ser imprescindível que o sistema armazene tanto o número da nota fiscal emitida para pagamento quanto a data em que esta foi paga pelo cliente. Além de controlar os orçamentos e serviços realizados pela empresa, o sistema deverá ser capaz de emitir os seguintes relatórios:

1. Listagem de peças em estoque na empresa, contendo o código da peça, o nome, o fabricante, o valor unitário e a quantidade disponível;
2. Relação mensal de gastos com os mecânicos, mostrando a matrícula, o nome, o telefone e o salário de cada um deles;
3. Listagem completa dos serviços realizados por cada mecânico no mês, exibindo o nome e a matrícula do mecânico, bem como o número e o valor dos serviços executados.

Modelo conceitual produzido a partir do minimundo 03



Esquemas relacionais transformando cada entidade numa tabela:

CLIENTE(CPF, Nome, Endereço, Telefone)

ORÇAMENTO(NúmeroOrçamento, Valor, Data, Descrição, CPF)

MECÂNICO(Matrícula, Nome, Telefone, Salário)

SERVIÇO(NúmeroServiço, Valor, Data, Descrição, CPF, Nota Fiscal, Data Pagamento,NúmeroOrçamento)

PEÇA(Código, Nome, Quantidade, ValorUnitário, Fabricante)

EXECUTA(NúmeroServiço, Matrícula)

USA(NúmeroServiço, Código Peça)

LISTA(NúmeroOrçamento, Código Peça)

Fim da atividade online

Alterando uma tabela

A SQL também dá a opção de alterar tabelas já definidas no banco de dados. Para isso utilizamos o comando ALTER TABLE.

Esse comando pode ser utilizado em conjunto com cláusulas adicionais para:

- incluir novas colunas numa tabela existente;
- adicionar a definição de uma restrição a uma tabela existente;
- excluir uma tabela.

Veja agora os dois primeiros itens. O último item será explicado mais adiante.

Incluindo novas colunas numa tabela existente

Para incluir novas colunas numa tabela, utilizamos o comando ALTER TABLE, conforme apresentado a seguir:

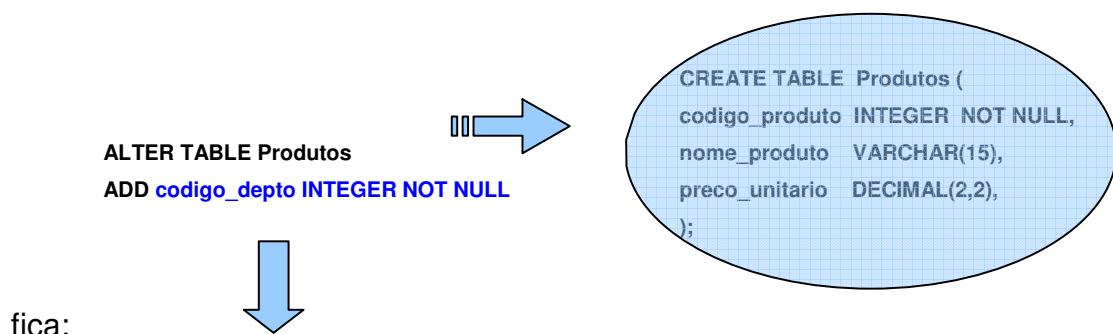
```
ALTER TABLE nome_tabela
ADD coluna nome_coluna tipo_coluna restrições
```

onde:

- ALTER TABLE: refere-se ao comando de alteração numa tabela;
- nome_tabela: nome da tabela que está sendo alterada;
- ADD: indica que uma nova coluna está sendo incluída na tabela;
- nome_coluna tipo_coluna restrições: nome da coluna seguido do tipo de dado e possíveis restrições.

Exemplo:

Adicionando a coluna código de departamento à tabela produtos (veja a Figura 7.10):



```
CREATE TABLE Produtos (
    codigo_produto INTEGER NOT NULL,
    nome_produto VARCHAR(15),
    preco_unitario DECIMAL(2,2),
    codigo_depto INTEGER NOT NULL,
);
```

Adicionar a definição de uma restrição a uma tabela existente

Para incluir uma nova restrição a uma tabela existente, também utilizamos o comando ALTER TABLE. A sintaxe básica a ser utilizada é:

```
ALTER TABLE nome_tabela
ADD CONSTRAINT nome_restricção tipo_restricção
                (coluna1_restricção, coluna2_restricção,..)
```

onde:

- ALTER TABLE: determina um comando de alteração em uma tabela;
- *nome_tabela*: nome da tabela que está sendo alterada;
- ADD CONSTRAINT: indica que uma nova restrição está sendo incluída na tabela;
- *nome_restricção*: nome da restrição que está sendo incluída na tabela;
- *tipo_restricção*: tipo da restrição que está sendo incluída, ou seja, PRIMARY KEY (chave primária), FOREIGN KEY (chave estrangeira) ou **UNIQUE**;
- *coluna1_restricção*, *coluna2_restricção*,..: representam a lista de colunas que participam da restrição.

Início do Verbete

UNIQUE

Restrição que indica que o campo ou o conjunto de campos de uma tabela deve possuir valores únicos ou distintos, sem repetições.

Fim do verbete

Exemplo:

Criamos a tabela Produtos sem especificar nenhuma restrição. Vamos agora adicionar a restrição de chave primária:

```
ALTER TABLE Produtos  
ADD CONSTRAINT PK_codigo PRIMARY KEY (codigo_produto)
```

Observamos que:

- nome da restrição que está sendo incluída: PK_codigo;
- tipo_restricão: PRIMARY KEY;
- coluna1_restricão: codigo_produto.

A chave primária poderia ter sido criada quando a tabela Produtos foi criada, da seguinte forma:

```
CREATE TABLE Produtos (  
    codigo_produto INTEGER NOT NULL,  
    nome_produto VARCHAR(15),  
    preco_unitario DECIMAL(2,2),  
    PRIMARY KEY (codigo_produto)  
)
```

Então, a chave primária é o primeiro campo da tabela: codigo_produto.

Excluindo uma tabela

Para excluir ou destruir uma tabela, utilizamos o comando DROP TABLE. Sua sintaxe básica é:

```
DROP TABLE nome_tabela Objetos_Opcional
```

onde:

- DROP TABLE: comando de destruição de uma tabela;
- *nome_tabela*: nome da tabela que está sendo destruída;

- **Objetos_Opcional:** indica que os objetos dependentes dessa tabela, tais como chaves estrangeiras que a referenciam, também devem ser excluídos.

Exemplo: Desejamos destruir a tabela produtos:

```
DROP TABLE Produtos
```

Início da atividade

Atividade 6 - Atende aos objetivos 2 e 3

Considere as tabelas Produto e Loja, apresentadas a seguir:

Tabela Produto

| codigo_produto | Descrição | Marca | Categoria | Preço Compra | Preço Venda |
|----------------|-----------|-------|-----------|--------------|-------------|
| | | | | | |

Tabela Loja

| codigo_loja | endereço | nome |
|-------------|----------|------|
| | | |

Utilizando os comandos de SQL, realize os seguintes exercícios:

- Crie duas tabelas.
- Adicione a coluna Nome do Responsável na tabela Loja.
- Adicione as chaves primárias em ambas as tabelas.
- Exclua a tabela Produto e a tabela Loja.

Diagramação: deixar 10 linhas para a resposta.

Resposta comentada

- Criando a tabela Produto e a tabela Loja:

```
CREATE TABLE Produto (
    codigo_produto INTEGER NOT NULL,
    nome_produto VARCHAR(15),
    preco_unitario DECIMAL(2,2),
);
```

```
CREATE TABLE Loja (
    codigo_loja INTEGER NOT NULL,
    endereço VARCHAR(15),
    nome DECIMAL(2,2),
);
```

b) Adicionando a coluna Nome do Responsável na tabela Loja.

```
ALTER TABLE Loja  
ADD nome_responsável VARCHAR(30)
```

c) Adicionando a chave primária na tabela Produto:

```
ALTER TABLE Produto  
ADD CONSTRAINT PK_codigo_produto PRIMARY KEY (codigo_produto)
```

Adicionando a chave primária na tabela Loja:

```
ALTER TABLE Loja  
ADD CONSTRAINT PK_codigo_loja PRIMARY KEY (codigo_loja)
```

d) Excluindo a tabela Produto:

```
DROP TABLE Produto
```

Excluindo a tabela Loja:

```
DROP TABLE Loja
```

Fim da atividade

Início da atividade online

Atividade 7 - Atende aos objetivos 3 e 4 (Diagramador, favor inserir ícone de atividade online)

Entre no ambiente virtual e resolva a atividade a seguir, enviando sua resposta ao tutor.

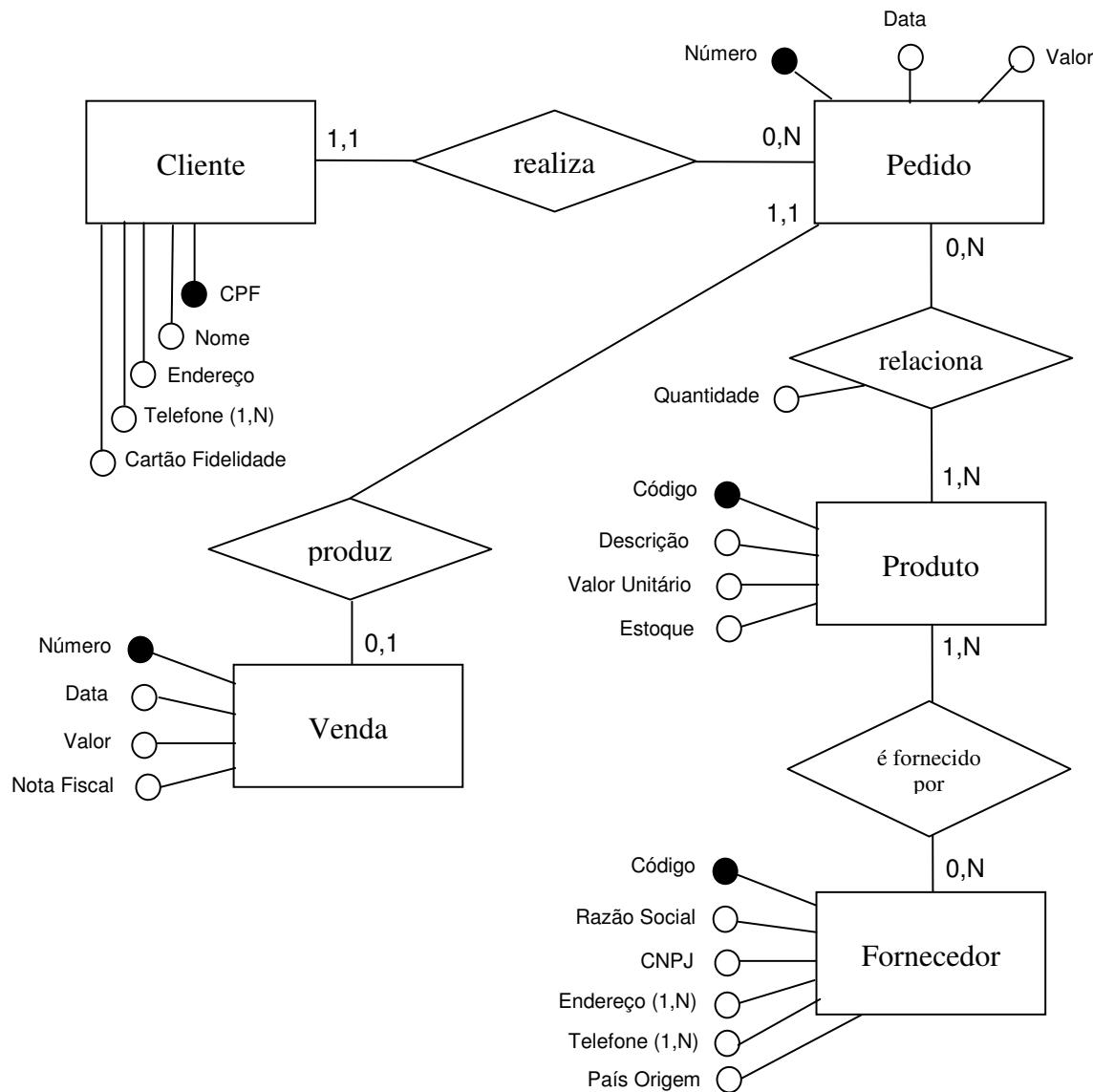
A partir do minimundo e dos modelos conceitual e lógico apresentados a seguir, descreva os comandos necessários para a criação do modelo físico correspondente e os comandos de criação de tabelas e chaves estrangeiras.

Minimundo 05

1. A empresa Importex é especializada na importação e venda local de produtos originados em outros países.
2. A empresa tem perdido várias vendas devido à falta do produto em estoque quando o cliente faz um pedido para comprá-lo. A diretoria da empresa decidiu, então, implantar um sistema que permita informar o nível de estoque dos produtos e monitorar as vendas para evitar novos prejuízos.
3. O usuário designado pela diretoria informou que, quando o cliente solicita um ou mais produtos, o funcionário verifica se eles existem em estoque. Em caso negativo, ele anota os produtos que devem ser comprados e sua respectiva quantidade de reposição. Se os produtos estiverem disponíveis, o atendente encaminha ao setor de vendas um pedido especificando os produtos e as quantidades solicitadas pelo cliente. Todo pedido possui um número único que o identifica. Os produtos são, então, separados e embalados. Simultaneamente, o setor de vendas emite uma nota fiscal que relaciona os produtos adquiridos, as respectivas quantidades e os preços de venda. O cliente efetua o pagamento no valor total da nota fiscal. Clientes com cartão de fidelidade têm direito a 10% de desconto sobre o total da nota.
4. O sistema a ser desenvolvido deve registrar cada venda efetuada, informando o número da venda (originado do pedido), o número da nota fiscal, a data e o valor total. A fim de saber se o cliente tem direito a desconto, a empresa mantém um cadastro de clientes onde constam CPF, endereço, nome, número do cartão de fidelidade e seus telefones. Da mesma forma, um cadastro de fornecedores lista todas as informações necessárias para contato no caso de aquisição de novos produtos.
5. Com o objetivo de evitar que novas vendas sejam perdidas devido à falta de produto em estoque, o sistema deverá ser capaz de emitir os seguintes relatórios:
 - a. Listagem de pedidos pendentes relacionando para cada um o seu número, data e valor;
 - b. Listagem de produtos sem estoque disponível, contendo código do produto, sua descrição e valor unitário;
 - c. Relatório das vendas diárias dos produtos, relacionando o número da nota fiscal, código, descrição do produto, data e valor da nota fiscal;

d. Listagem dos fornecedores de produtos ordenada por país de origem. A listagem deve exibir o código, a razão social, o CNPJ, o(s) endereço(s) e o(s) telefone(s) para contato.

Modelo conceitual produzido a partir do minimundo 05



Esquemas relacionais transformando cada entidade numa tabela:

CLIENTE (CPF, Nome, Endereço, Telefone, CartãoFidelidade)

PEDIDO (NúmeroPedido, Data, Valor, CPF)

PRODUTO (CódigoProduto, Descrição, ValorUnitário, Estoque)

VENDA (NúmeroVenda, Data, Valor, NotaFiscal, NúmeroPedido)

FORNECEDOR (CódigoFornecedor, RazãoSocial, CNPJ, Endereço, Telefone, PaísOrigem)

RELACIONA (NúmeroPedido, CódigoProduto, quantidade)

FORNECIDO (CódigoFornecedor, CódigoProduto)

Fim da atividade online

Conclusão

Após a criação do modelo lógico de banco de dados, partimos para a construção do modelo físico, ou seja, utilizamos as tabelas que foram criadas. Para isso, escolhemos o SGBD adequado e, com ajuda da linguagem SQL, especificamente DDL, podemos criar, alterar e excluir as tabelas, pois essa linguagem é uma poderosa ferramenta de manipulação de banco de dados. Dessa forma, com a transformação do modelo relacional para o modelo físico, o projeto de banco de dados ficará de acordo com aquilo que foi definido nos minimundos.

Resumo

Estes são os principais conceitos vistos nesta aula:

Descrição dos diferentes tipos de dados:

- INTEGER ou INT: número positivo ou negativo inteiro;
- NUMERIC: número positivo ou negativo de ponto flutuante. Normalmente, deve-se informar o tamanho total do campo e definir quantas casas decimais devem ser armazenadas após a vírgula;
- DECIMAL: semelhante ao NUMERIC, mas, em alguns bancos de dados, poderá ter maior precisão depois da vírgula;
- CHAR: permite armazenar cadeias de caracteres (letras, combinação de símbolos, letras e números). O tamanho deve ser informado e será fixo, ou seja, mesmo que não utilizado totalmente, o espaço será ocupado fisicamente;
- VARCHAR: permite armazenar cadeias de caracteres, mas com tamanho variável. Se for utilizado menos espaço que o máximo definido, o espaço

restante não será ocupado.

Outros tipos de dados:

- FLOAT: número de ponto flutuante de simples precisão. Não confundir com o DOUBLE, que também é utilizado para números de ponto flutuante, mas de dupla precisão, ou seja, maior aproximação de resultados;
- DATE: permite armazenar datas;
- TIME: permite armazenar horários.

Restrições de integridade do modelo físico

Dizer que os dados de um banco de dados estão íntegros significa dizer que eles refletem corretamente a realidade representada pelo banco de dados e que são consistentes entre si.

Tipos de restrições de integridade

- Integridade de domínio: o valor de um atributo deve pertencer ao domínio;
- Integridade de vazio: para cada atributo deve-se definir se ele pode assumir o valor nulo;
- Integridade de chave: refere-se à definição das chaves primárias e alternativas, indicando que os atributos ou o conjunto de atributos devem ter valores únicos;
- Integridade referencial: indica que os valores dos atributos numa chave estrangeira devem aparecer na chave primária de uma tabela referenciada. As restrições de integridade referencial surgem dos relacionamentos entre as entidades representadas em esquema de relação.

Linguagem de Definição de Dados (DDL)

DDL (*Data Definition Language*) – Linguagem utilizada para definir a estrutura de armazenamento dos dados, também chamada de dicionário de dados. Através dela podemos criar, visualizar, alterar e excluir uma tabela.

Criando uma tabela

Ao criarmos uma tabela, devemos especificar:

- seu nome;
- as colunas que a compõem;
- os tipos de dados das colunas;
- se uma coluna refere-se à chave primária ou não; e
- as regras de integridade, entre outros.

Uma coluna definida como de tipo INTEGER pode suportar números inteiros ou o valor NULL. Da mesma forma, uma coluna CHAR suporta cadeias de caracteres e o valor NULL. Lembre-se que NULL indica ausência de valor definido; é diferente de zero. Aquelas colunas que não podem conter o valor NULL têm preenchimento obrigatório.

Alterando uma tabela

A SQL dá a opção de alterar tabelas já definidas no banco de dados. Para isso, utilizamos o comando ALTER TABLE.

Este comando pode ser utilizado em conjunto com cláusulas adicionais para:

- incluir novas colunas numa tabela existente;
- adicionar a definição de uma restrição a uma tabela existente;
- excluir uma tabela.

Incluindo novas colunas numa tabela existente

Para incluir novas colunas numa tabela, utilizamos o comando ALTER TABLE.

Adicionar a definição de uma restrição a uma tabela existente

Para incluir uma nova restrição a uma tabela existente, também utilizamos o comando ALTER TABLE.

Excluindo uma tabela

Para excluir ou destruir uma tabela utilizamos o comando DROP TABLE.

Informações sobre a próxima aula

Na próxima aula, você verá a utilização de índices que servem para aumentar o desempenho em operações de leituras no banco de dados. Também estudaremos os comandos DML (*data manipulation language*), inserindo, atualizando e removendo dados de uma tabela de um banco de dados.

Referências bibliográficas

- DATE, C. J. *Introdução a sistemas de bancos de dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.
- CARVALHO, C. R. *SQL - Guia prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.
- ELMASRI R.; NAVATHE, S. *Sistemas de banco de dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.
- HEUSER, C. A. *Projeto de banco de dados*. 4^a ed. Porto Alegre: Bookman, 2009.
- SETZER, V. W.; CORRÊA DA SILVA, F. S. *Bancos de dados*. São Paulo: Edgard Blucher, 2005.
- SILBERSCHATZ, A.; KORTH, H. *Sistema de banco de dados*. 3^a ed. São Paulo: Pearson Makron Books, 2008.

Disciplina de Extensão: Modelando e implementando bancos de dados relacionais
Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 8

Mapeamento do modelo relacional para o físico (Parte 2 - Criação de índices e manipulação de dados)

Meta

Apresentar a continuação do mapeamento do modelo relacional para o modelo físico, considerando características desejadas de desempenho, como a criação de índices e manipulação de dados utilizando a linguagem padrão SQL.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. identificar os tipos de índices;
2. criar, alterar e excluir um índice;
3. representar e aplicar a integridade referencial;
4. descrever os comandos da linguagem de manipulação de dados DML para:
 - inserir dados em uma tabela (INSERT);
 - atualizar dados em uma tabela (UPDATE);
 - remover dados de uma tabela (DELETE).

Pré-requisitos

Para um bom aproveitamento desta aula, é importante você relembrar os conceitos de tipos de dados e criação de tabela, estudados na Aula 7.

A importância dos índices!

Você já parou para pensar o quanto um índice é importante?

Imagine que você compre um livro de 500 páginas e ele não apresente índice

mostrando o seu conteúdo. Para qualquer pesquisa no livro você precisará procurar em todo o livro... Já imaginou o trabalho que isso daria? Já um livro que possui índice permite ir direto ao ponto que interessa.



Figura 8.1: Procurar um determinado assunto em um livro sem índice? Tarefa complicada...

Livro: Fonte: <http://www.sxc.hu/photo/1170739> | Foto: sanja gjenero

Homem: Fonte: <http://www.sxc.hu/photo/418215> | Foto: Bob Smith

Agora imagine as aplicações bancárias que atendem aos caixas eletrônicos. Sempre que solicitamos uma determinada transação ou informação, tal solicitação tende a ser rapidamente atendida. Se não fosse pela utilização dos índices, uma simples pesquisa pelo seu saldo demoraria muito para retornar uma resposta à sua solicitação.



Figura 8.2: Uma simples consulta ao saldo bancário seria uma tarefa bem trabalhosa sem o índice.

Fonte: <http://www.sxc.hu/photo/794351> | Foto: Vaughan Willis

Uma vez tendo ciência do funcionamento dos índices e respeitando a sua regra de negócios, uma consulta deverá ter resposta em tempo satisfatório.

Então, um índice é uma ferramenta que permite acessar rapidamente o local do dado procurado e, assim, acelerar as buscas ou pesquisas.

Por outro lado, a linguagem SQL (*Structured Query Language*) é a base para utilização de bancos de dados relacionais. Com a utilização dos comandos básicos (INSERT, UPDATE e DELETE) pode-se resolver a maior parte dos problemas relacionados à manutenção e à extração de dados no banco de dados. Com os comandos SQL é possível criar as estruturas básicas de armazenamento, como tabelas e índices. E esta aula trata justamente desses tipos de conceitos. Vamos nessa!

Fim da Introdução

Índice

Índices são utilizados para aumentar o desempenho em operações de leitura no banco de dados. O índice serve para prover acesso rápido às linhas das tabelas. Por meio dele, é possível unir uma ou mais colunas por onde o acesso é mais frequente. Por exemplo: temos uma tabela de funcionários contendo os seguintes dados:

| Matrícula | Nome | Endereço | Sexo | Salário |
|-----------|-----------------------|-------------------|------|----------|
| 10101 | Maria José Silva | Rua Ouros, 123 SP | F | 1.233,00 |
| 10109 | Pedro Oliveira Campos | Rua E, 111 SP | M | 1.469,00 |
| 10111 | Priscilla dos Santos | Rua Rica, 34 SP | F | 1.221,00 |
| 10231 | Jorge Bravo | Rua Verde, 12 SP | M | 1.561,00 |
| 10244 | Maria Campos Silva | Rua Campo, 678 SP | F | 1.121,00 |

Muito tempo
para procurar
linha por linha

Figura 8.3. Tabela de funcionários.

Se desejamos fazer buscas, usar o nome dos funcionários como chave primária não

seria uma boa (pela possível repetição de nomes). Então a chave primária é a matrícula dos funcionários. Além da chave primária, que é um identificador único dos funcionários, você pode também criar um índice para colunas que poderá auxiliá-lo nas consultas.

Assim, mesmo que o índice não seja a chave primária, garante um acesso mais rápido aos nomes.

A localização do dado procurado pode ser acessada rapidamente

↓

Índice →

| Matrícula | Nome | Endereço | Sexo | Salário |
|-----------|-----------------------|-------------------|------|----------|
| 10101 | Maria José Silva | Rua Ouros, 123 SP | F | 1.233,00 |
| 10109 | Pedro Oliveira Campos | Rua E, 111 SP | M | 1.469,00 |
| 10111 | Priscilla dos Santos | Rua Rica, 34 SP | F | 1.221,00 |
| 10231 | Jorge Bravo | Rua Verde, 12 SP | M | 1.561,00 |
| 10244 | Maria Campos Silva | Rua Campo, 678 SP | F | 1.121,00 |

Figura 8.4. Tabela de funcionários e índice.

Quando um índice é criado, o banco de dados cria uma tabela à parte contendo as colunas que compõem o índice e coloca os dados de forma sequencial. Dessa forma, toda vez que uma informação é procurada o banco de dados aproveita o índice que já está ordenado e realiza a pesquisa com muito mais agilidade, utilizando os dados do índice (tabela ordenada), ao invés de buscar diretamente na tabela original.

Tipos de índices

Índice comum (**INDEX**):

➤ Estes índices são criados a partir de colunas que não pertencem às chaves primárias de tabelas. Podem ser criados usando quaisquer colunas da tabela. Permitem inclusive que a coluna indexada possua repetições de valores. São utilizados para agilizar buscas realizadas em cima de campos que não estão inclusos na chave primária da tabela.

Chave primária (**PRIMARY KEY**)

➤ A chave primária é um tipo especial de índice que garante a singularidade: os valores nas colunas de chave primária são únicos ao longo de todas as linhas da tabela.

Uma tabela pode ter apenas uma chave primária, e nenhum dos valores das colunas na chave primária pode ser anulável;

- Toda chave primária deve ser definida;
- MySQL requer que se especifique NOT NULL nas colunas que serão utilizadas como chave, uma vez que as colunas que compõem a chave primária não podem conter valores nulos ou repetidos.

Índice único (**UNIQUE**):

- Implica que não pode haver repetições na coluna ou combinação de colunas;
- É similar à restrição de chave primária, com duas exceções: um índice único pode conter colunas anuláveis e uma tabela pode ter vários índices únicos. Cada índice único pode envolver uma única coluna ou ainda mais de uma coluna da tabela, estejam elas combinadas ou não;
- É usado em chaves primárias.

Índice múltiplo

- É um índice que agrupa mais de um campo da tabela;
- É usado quando uma consulta deve ser executada utilizando ao mesmo tempo mais de um atributo como filtro.

Início da atividade

Atividade 1 - Atende ao objetivo 1

1. Observe a tabela de funcionários a seguir e identifique, para cada uma das colunas, os tipos de índice que podem ser criados.

| Matricula | Nome | Endereço | CEP | Sexo | Salário |
|-----------|-----------------------|-------------------|-------|------|----------|
| 10101 | Maria José Silva | Rua Ouros, 123 SP | 22810 | F | 1.233,00 |
| 10109 | Pedro Oliveira Campos | Rua E, 111 SP | 21312 | M | 1.469,00 |
| 10111 | Priscilla dos Santos | Rua Rica, 34 SP | 27493 | F | 1.221,00 |
| 10231 | Jorge Bravo | Rua Verde, 12 SP | 21283 | M | 1.561,00 |
| 10244 | Maria Campos Silva | Rua Campo, 678 SP | 23810 | F | 1.121,00 |

2. Se você fosse criar um índice envolvendo as colunas endereço e CEP, qual seria o tipo desse índice?

Diagramação, inserir 5 linhas para a resposta.

Resposta Comentada

1. *Campo matrícula – índice do tipo chave primária;*

Campos Nome, Endereço, CEP e Salário – Índice do tipo único;

Campo Sexo – Índice do tipo comum;

2. *Combinando as colunas Endereço e CEP poderia ser criado um índice do tipo múltiplo.*

Fim da atividade

Os índices deverão ser usados quando for necessário, pois requerem mais espaço no disco. Existem critérios para a criação de índices, pois, em grande quantidade, tornam mais lentas as inserções, exclusões e atualizações nos campos da tabela. Assim, a manutenção e o monitoramento são de extrema importância, evitando que os índices se tornem obsoletos e ocupem espaço desnecessário no computador.

Contribuindo para uma melhor performance, agora você vai aprender como criar, alterar e excluir índices.

Criando um índice

A criação de índices para campos de uma tabela é o principal modo de otimização de acesso a dados fornecidos por MySQL. A busca por dados indexados pode ser de muitas ordens de magnitude mais rápidas que a busca por dados não-indexados.

Para especificar os campos indexados na criação da tabela, usa-se a palavra-chave **INDEX** dentro da definição dos campos da tabela.

```

CREATE TABLE nome_tabela
(
    nome_campo1,
    nome_campo2,
    nome_campo3, ...
    INDEX (nome_campo_a_ser_indexado)
)

```

Se a tabela já existe e você precisa criar um índice, a sintaxe básica é:

```

CREATE UNIQUE INDEX nome_idx
ON nome_tabela (nome_coluna1, ...) ASC/DESC

```

onde:

UNIQUE: identifica que este índice não permite repetição na chave.

nome_idx : nome do índice a ser criado.

nome_tabela: nome da tabela onde o índice será criado.

nome_coluna1, ... : lista de colunas que compõem a chave de indexação.

ASC: determina que a ordem de indexação seja ascendente.

DESC: determina que a ordem de indexação seja descendente.

Por outro lado, a cada campo do arquivo de dados corresponde um campo no arquivo índice. O índice pode estar ordenado, apesar de o arquivo de dados não estar. Em geral, o arquivo de dados está organizado segundo a ordem de entrada dos registros.

Exemplo: Imagine que temos a tabela Artigos com as seguintes informações:

| Código identificação | Dpto. identificação | Tipo | Data lançamento |
|----------------------|---------------------|--------|-----------------|
| 22311 | 101 | CD | 12/01/09 |
| 22112 | 104 | miniCD | 08/04/09 |
| 212101 | 102 | DVD | 11/09/09 |

Figura 8.5. Tabela de artigos de produtos.

Vamos criar a tabela e o índice utilizando o comando INDEX dentro da criação da

tabela da seguinte forma:

```
CREATE TABLE Artigos (
    codigo_ident INT PRIMARY KEY,
    depto_ident INT NOT NULL,
    tipo VARCHAR(20) NOT NULL,
    data_lançamento DATE NOT NULL,
    INDEX (codigo_ident)
);
```

Assim, criamos a tabela denominada Artigos com seus campos e criamos o índice chamado código_ident, campo a ser indexado na utilização de busca mais rápida por aplicações e usuários.

Agora suponha que a tabela não existe e desejamos somente criar essa tabela utilizando os comandos SQL:

```
CREATE TABLE Artigos (
    codigo_ident INT PRIMARY KEY,
    depto_ident INT NOT NULL,
    tipo VARCHAR(20) NOT NULL,
    data_lançamento DATE NOT NULL
);
```

Pela procura constante das informações do código dos itens, desejamos criar um índice chamado código_ident_index; então executamos os seguintes comandos:

```
CREATE UNIQUE INDEX código_ident_index
ON Artigos (codigo_ident) ASC
```

Diagrama explicativo dos componentes do comando de criação de índice:

- CREATE UNIQUE INDEX**: Nome do comando.
- código_ident_index**: Nome do índice, circulado em amarelo.
- ON Artigos**: Tabela, circulado em azul.
- (codigo_ident)**: nome coluna ou campo.
- ASC**: Ordenação ascendente.
- nomes índice**: Destino final do comando.

Assim, em cada consulta dos itens dos artigos, a aplicação poderá mostrar os códigos dos artigos em forma ascendente.

Alterando um índice

Para transformar em índice um campo já existente, usa-se a sintaxe de ALTER TABLE:

```
ALTER TABLE nome_tabela ADD INDEX  
nome_campos;
```

Excluindo um índice

Para remover um índice, basta usar DROP INDEX:

```
DROP INDEX nome_tabela  
nome Índice;
```

Início da atividade

Atividade 2 - Atende ao objetivo 2

Suponha que temos uma enorme coleção de CDs, e desejamos acessá-los através de um arquivo. Para cada registro, mantemos as seguintes informações:

- id_numero: número de identificação,
- nome_titulo: o título do CD,
- compositor: nome do(s) compositor(es),
- artista: nome do(s) artista(s),
- id_gravadora: código da gravadora.

- a) Como organizar esse arquivo para garantir acesso rápido a um registro qualquer, dada a sua chave primária id_numero?
- b) Num determinado momento, observamos que não é preciso manter o índice do número de identificação, pois está tendo mais consultas dos usuários nos outros campos do arquivo. Então realize a exclusão desse índice.
- c) Considerando o item a, deseja-se mudar o índice número de identificação para código de gravadora. Como deve ser feita essa troca de índice?

Diagramação, inserir 10 linhas para a resposta.

Resposta comentada

a) A chave primária para os registros está formada pelo campo `id_numero`; isso garante que cada registro esteja unicamente associado à sua chave primária. Poderíamos pensar em construir a tabela juntamente com o índice, pois ela não existe. Então, temos os seguintes comandos:

```
CREATE TABLE Artigos (
    id_numero INT NOT NULL PRIMARY KEY,
    nome_titulo VARCHAR(10) INT NOT NULL,
    compositor VARCHAR(35) NOT NULL,
    artista VARCHAR(35) NOT NULL,
    id_gravadora INT,
    INDEX (id_numero)
);
```

A estrutura do índice pode ser muito simples: a cada campo do arquivo de dados corresponde um campo no arquivo índice.

b) Para remover o índice do número de identificação, basta usar `DROP INDEX` da seguinte maneira:

```
DROP INDEX Artigos
    id_numero;
```

c) Considerando que o índice é o número de identificação e se deseja mudar o índice para o campo de código de gravadora, então utilizamos a seguinte sintaxe:

```
ALTER TABLE Artigos ADD INDEX
    id_gravadora;
```

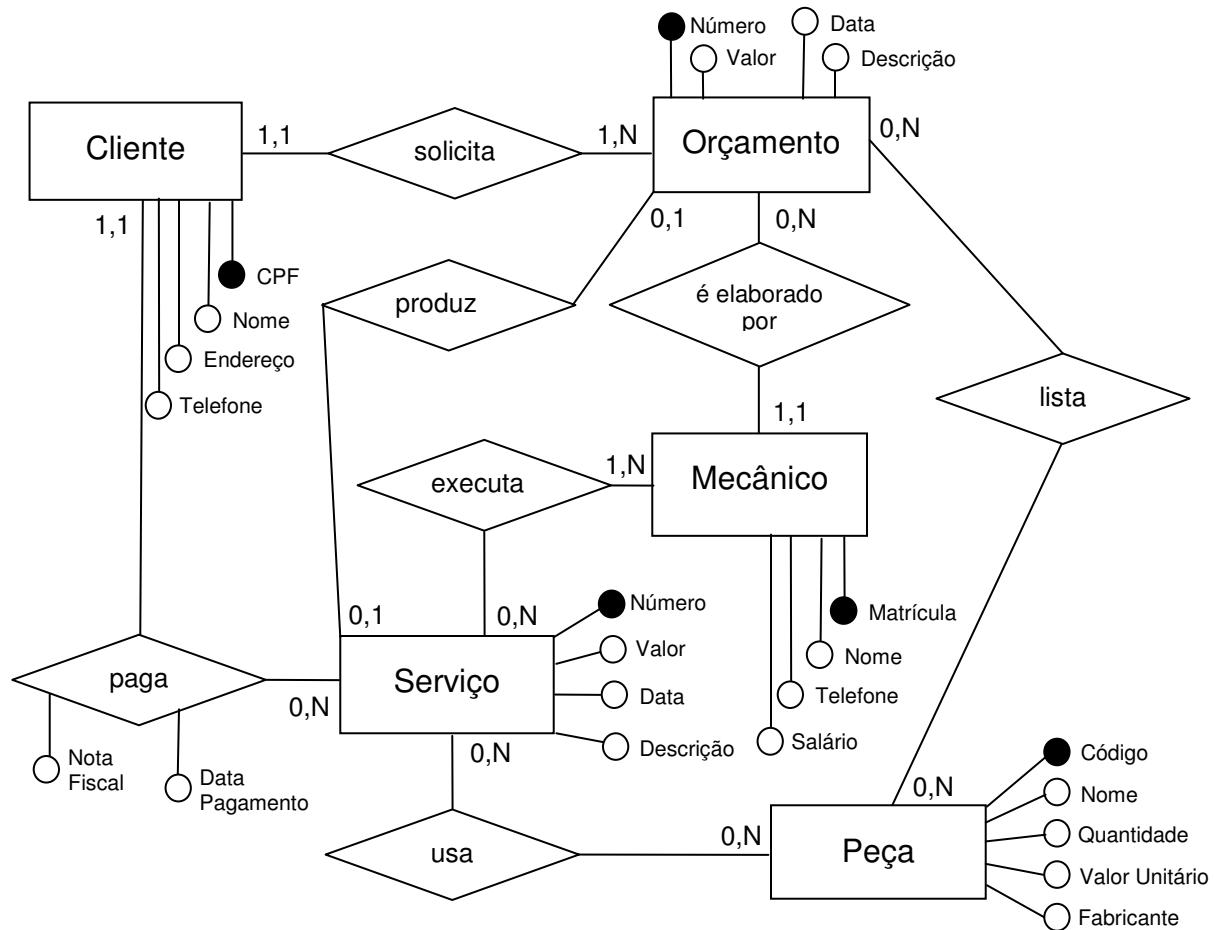
Fim da atividade

Início da atividade 3

Atividade 3 - Atende ao objetivo 2

A partir do modelo ER apresentado, crie os esquemas relacionais e, em seguida, liste os comandos necessários para:

- Criação de tabelas com índices;
- Criação de índices supondo que as tabelas já existam.



Resposta Comentada

1. Criação dos esquemas relacionais:

A tabela **CLIENTE** é criada a partir da entidade **CLIENTE** usando os seus atributos:
CLIENTE (CPF, *Nome*, *Endereço*, *Telefone*);

A tabela **ORÇAMENTO** é criada a partir da entidade **ORÇAMENTO** usando os seus atributos e acrescentando:

- O campo **CPF**, em função do relacionamento **1xN** com a entidade **CLIENTE**;
- O campo **Matricula**, em função do relacionamento **1xN** com a entidade **MECÂNICO**:
ORÇAMENTO (Número, *Valor*, *Data*, *Descrição*, *CPF*, *Matricula*);

A tabela **SERVIÇO**, criada apenas com o atributo **NumeroOrçamento**, em função do

relacionamento 1x1 existente com a entidade ORÇAMENTO. Os demais atributos são iguais e já estão na tabela ORÇAMENTO, não sendo necessário acrescentá-los.

Também são acrescentados os campos NotaFiscal e DataPagamento, que são originados do relacionamento 1xN da entidade SERVIÇO com a entidade CLIENTE.

SERVIÇO (NúmeroOrçamento, NotaFiscal, DataPagamento);

A tabela MECÂNICO é criada a partir da entidade MECÂNICO usando os seus atributos:

MECÂNICO (Matrícula, Nome, Telefone, Salário);

A tabela PEÇA é criada a partir da entidade PEÇA usando os seus atributos:

PEÇA (Código, Nome, Quantidade, Valor Unitário, Fabricante);

A tabela LISTA é criada a partir do relacionamento NxN entre as entidades ORÇAMENTO e PEÇA, possuindo como campos os atributos-chave de ambas as entidades.

LISTA (CódigoPeça, NúmeroOrçamento);

A tabela EXECUTA é criada a partir do relacionamento NxN entre as entidades MECÂNICO e SERVIÇO, possuindo como campos os atributos-chave de ambas as entidades.

EXECUTA (MatriculaMecânico, NúmeroServiço);

A tabela USA é criada a partir do relacionamento NxN entre as entidades SERVIÇO e PEÇA, possuindo como campos os atributos-chave de ambas as entidades.

USA (CódigoPeça, NúmeroServiço);

a) Criação das tabelas com índices:

Esquema relacional das tabelas

CLIENTE (CPF, Nome, Endereço, Telefone);

ORÇAMENTO (Número, Valor, Data, Descrição, CPF, Matricula);

*SERVIÇO (NúmeroOrçamento, NotaFiscal, DataPagamento);
MECÂNICO (Matrícula, Nome, Telefone, Salário);
PEÇA (Código, Nome, Quantidade, Valor Unitário, Fabricante);
LISTA(CódigoPeça, NúmeroOrçamento);
EXECUTA(MatriculaMecânico, NúmeroServiço);
USA (CódigoPeça, NúmeroServiço);*

Criação da tabela CLIENTE com índice:

```
CREATE TABLE CLIENTE (
        CPF VARCHAR(11) NOT NULL PRIMARY KEY,
        Nome VARCHAR(60) NOT NULL,
        Endereco VARCHAR(100) NOT NULL,
        Telefone VARCHAR(20) NOT NULL,
        INDEX (CPF)
);
```

Criação da tabela ORÇAMENTO com índice:

```
CREATE TABLE ORCAMENTO (
        Numero INT NOT NULL PRIMARY KEY,
        Valor NUMERIC(9,2) NOT NULL,
        Data DATETIME NOT NULL,
        Descricao VARCHAR(500),
        CPF VARCHAR(11) NOT NULL,
        Matricula INT,
        INDEX (Numero)
);
```

Criação da tabela SERVIÇO com índice:

```
CREATE TABLE SERVIÇO (
        NumeroOrcamento INT NOT NULL PRIMARY KEY,
        NotaFiscal VARCHAR(20),
```

```
DataPagamento DATETIME,
INDEX (NumeroOrcamento)
);
```

Criação da tabela MECÂNICO com índice:

```
CREATE TABLE MECANICO (
    Matricula INT NOT NULL PRIMARY KEY,
    Nome VARCHAR(60) NOT NULL,
    Telefone VARCHAR(20) NOT NULL,
    Salario NUMERIC(9,2) NOT NULL,
    INDEX (Matricula)
);
```

Criação da tabela PEÇA com índice:

```
CREATE TABLE PECA (
    Codigo INT NOT NULL PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Quantidade INT NOT NULL,
    ValorUnitario NUMERIC(9,2),
    Fabricante VARCHAR(100),
    INDEX (Codigo)
);
```

Criação da tabela LISTA com índice:

```
CREATE TABLE LISTA (
    CodigoPeca INT NOT NULL,
    NumeroOrcamento INT NOT NULL,
    PRIMARY KEY (CodigoPeca, NumeroOrcamento),
    INDEX (CodigoPeça, NumeroOrcamento)
);
```

Criação da tabela EXECUTA com índice:

```
CREATE TABLE EXECUTA (
    MatriculaMecanico INT NOT NULL,
    NumeroServico INT NOT NULL,
    PRIMARY KEY (MatriculaMecanico, NumeroServico),
    INDEX (MatriculaMecanico, NumeroServico)
);
```

Criação da tabela USA com índice:

```
CREATE TABLE USA (
    CodigoPeca INT NOT NULL,
    NumeroServico INT NOT NULL,
    PRIMARY KEY (CodigoPeca, NumeroServico),
    INDEX (CodigoPeca, NumeroServico)
);
```

b) criação dos índices supondo que as tabelas já existam:

Esquema relacional das tabelas

*CLIENTE (CPF, Nome, Endereço, Telefone);
ORÇAMENTO (Número, Valor, Data, Descrição, CPF, Matricula);
SERVIÇO (NúmeroOrçamento, NotaFiscal, DataPagamento);
MECÂNICO (Matrícula, Nome, Telefone, Salário);
PEÇA (Código, Nome, Quantidade, Valor Unitário, Fabricante);
LISTA (CódigoPeça, NúmeroOrçamento);
EXECUTA (MatriculaMecânico, NúmeroServiço);
USA (CodigoPeça, NúmeroServiço);*

Comandos para a criação dos índices:

```
ALTER TABLE CLIENTE ADD UNIQUE INDEX ix_cliente (CPF);
ALTER TABLE ORCAMENTO ADD UNIQUE INDEX ix_orcamento (Número);
ALTER TABLE SERVICO ADD UNIQUE INDEX ix_servico (NúmeroOrcamento);
```

```

ALTER TABLE MECANICO ADD UNIQUE INDEX ix_mecanico (Matricula);
ALTER TABLE PECA ADD UNIQUE INDEX ix_peca (Codigo);
ALTER TABLE LISTA ADD UNIQUE INDEX ix_lista (CodigoPeca, NumeroOrcamento);
ALTER TABLE EXECUTA ADD UNIQUE INDEX ix_executa (MatriculaMecanico,
    NumeroServiço);
ALTER TABLE EXECUTA ADD UNIQUE INDEX ix_usa (CodigoPeca, NumeroServiço);

```

Fim da Atividade 3

Integridade referencial

O conceito de integridade referencial refere-se à manutenção automática da consistência das referências cruzadas entre diferentes tabelas.

A integridade referencial:

- garante o sincronismo de valores entre a chave estrangeira (*foreign key*) de uma tabela filha (tabela cuja chave estrangeira faz referência à chave primária da tabela pai) e a respectiva chave primária da tabela pai;
- propaga atualizações e exclusões efetuadas na tabela pai para as tabelas filhas, em cascata.

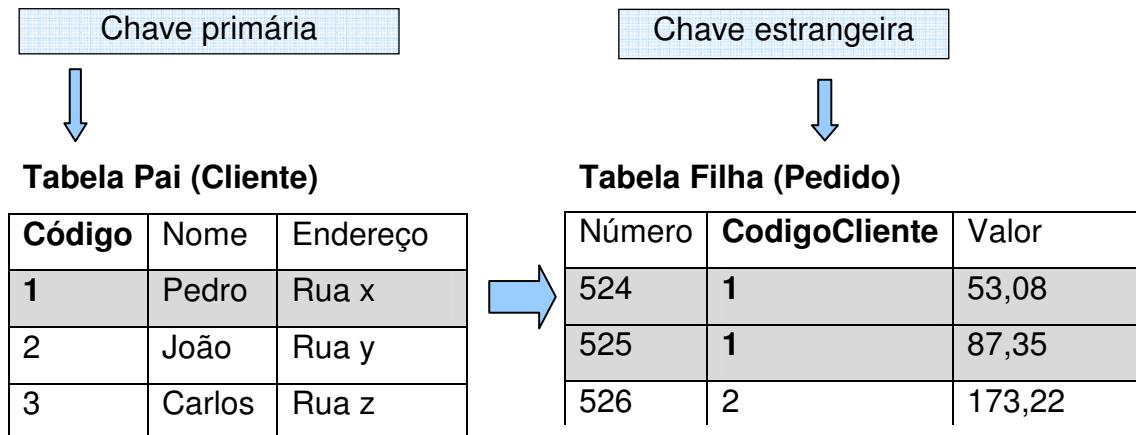


Figura 8.6. Sincronismo entre a tabela pai e tabela filha.

Chave estrangeira

A indicação explícita de chaves estrangeiras ajuda a manter a consistência na inserção de novas linhas, atualização de campos e exclusão de linhas. Para especificar que um

campo é uma chave estrangeira, usa-se a seguinte sintaxe no final da criação da tabela:

```
.....  
nome_coluna_tabela_filha tipo_dado FOREIGN KEY,  
REFERENCES nome_tabela_pai.nome_coluna,
```

Usam-se estes comandos no final da tabela.

sendo:

FOREIGN KEY: chave estrangeira, nome da coluna da tabela filha;

É necessário definir para o banco de dados as colunas que são chaves estrangeiras.

Trata-se dos campos que são chaves primárias de outras tabelas.

REFERENCES nome_tabela_pai.nome_coluna: na opção REFERENCES deve ser especificado o nome e a coluna da chave primária da tabela pai.

Exemplo: Imagine que temos a seguinte tabela de Funcionário:

| Matricula | Deptº | Nome | Data Entrada | Sexo | Salário |
|-----------|-------|----------------------|--------------|------|---------|
| 2191101 | DACC | João Pedro Silva | 01/12/00 | M | 567,00 |
| 2111012 | DMMA | Lucas Campos Santos | 12/01/09 | M | 742,30 |
| 2110113 | DAMM | Maria dos Santos | 06/09/08 | F | 872,00 |
| 2110237 | DMNI | Clemente Rosa Campos | 01/03/03 | M | 677,00 |
| 2110247 | DMAF | Lena Melo Oliveira | 05/11/06 | M | 580,00 |

Figura 8.7. Tabela Funcionário – Tabela pai

Utilizando a linguagem SQL, vamos criar a tabela chamada Funcionário, considerando como chave primária o campo matrícula:

```
CREATE TABLE Funcionário(  
matricula INT NOT NULL PRIMARY KEY,  
depto_ident CHAR(4) INT NOT NULL,  
nome_func VARCHAR(50) NOT NULL,  
data_entrada DATE NOT NULL,  
sexo CHAR(1),  
salario DECIMAL(3,2)  
);
```

Agora considere a tabela Dependente:

| Código Dependente | Nome Dependente | Data Nascimento |
|-------------------|--------------------------|-----------------|
| 2191101 | Carlos Campos Silva | 01/12/00 |
| 2111012 | Lúcia Campos Silva | 12/01/09 |
| 2110113 | Celeste dos Santos | 06/09/08 |
| 2110237 | Clemente Oliveira Junior | 01/03/03 |
| 2110247 | Rita Sifuentes Pereira | 05/11/06 |

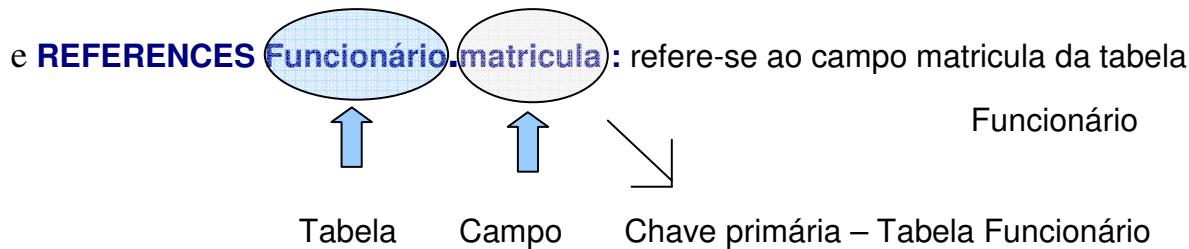
Figura 8.8. Tabela Dependente – Tabela filho

Criaremos a tabela chamada Dependente utilizando os comandos de SQL, considerando a chave primária código de dependente. Logo, declaramos a integridade referencial na tabela Dependente no campo matricula_funcionário; esse campo deve ser criado e definido como chave estrangeira. Vejamos:

```
CREATE TABLE Dependente  
  (código_dependente INT NOT NULL PRIMARY KEY,  
   nome_dependente VARCHAR(50),  
   data_nascimento DATE,  
   matricula_funcionário INT FOREIGN KEY,  
   REFERENCES Funcionário.matrícula  
 );
```

↓ ↓
Tabela chave primária

O comando matrícula_funcionário INT **FOREIGN KEY**, produzirá na tabela Dependente mais uma coluna, o campo matrícula funcionário, pois ele foi declarado chave estrangeira.



Com essa declaração, o SGBD garante que a inclusão de um DEPENDENTE somente será válida caso exista o FUNCIONÁRIO correspondente.

Atualização e exclusão de campos relacionados

```

nome_coluna_tabela_filha tipo_dado FOREIGN KEY,
REFERENCES nome_tabela_pai.nome_coluna,
ON UPDATE ação,
ou ON DELETE ação,

```

sendo:

ON DELETE: esta opção especifica os procedimentos que devem ser feitos pelo SGBD no caso da exclusão de um registro na tabela pai quando existe um registro correspondente nas tabelas filhas.

ON UPDATE: Esta opção especifica os procedimentos que devem ser feitos pelo SGBD quando houver uma atualização ou modificação de um registro na tabela pai, caso exista um registro correspondente na tabela filhas.

Ação refere-se a:

NO ACTION (sem ação): alterações que causam inconsistências são rejeitadas (padrão). Se alguma linha relacionada ainda existe quando a restrição é verificada, um erro aparecerá; este é o comportamento padrão se nada for especificado (a diferença essencial entre duas escolhas é que “sem ação” permite que a verificação seja adiada até depois na transação, e a “restrição” não.)

CASCADE: se a linha da tabela pai for apagada/modificada, a linha correspondente na tabela filha também será apagada, pois esta opção realiza a exclusão em todas as

tabelas filhas que possuam o valor da chave que será excluída na tabela pai.

SET NULL: se a linha da tabela pai for apagada/modificada, a linha correspondente na tabela filha receberá NULL, pois esta opção atribui o valor NULO às colunas das tabelas filhas que contenham o valor da chave que será excluída na tabela pai.

Exemplo: utilizando as tabelas 8.7 e 8.8, vamos indicar que sempre que for excluído um registro da tabela Funcionário, o SGBD deve excluir automaticamente os registros da tabela Dependente a ele relacionados. Os comandos em SQL são:

```
CREATE TABLE Dependente
(código_dependente INT NOT NULL PRIMARY KEY,
nome_dependente VARCHAR(50),
data_nascimento DATE,
matrícula_funcionário INT FOREIGN KEY,
REFERENCES Funcionário.matricula)
ON DELETE CASCADE
```

Neste caso, a cláusula **ON DELETE CASCADE** indica que, se um registro da tabela pai (Funcionário) for apagado, o registro correspondente na tabela filha (Dependente) também será apagado.

Início da atividade

Atividade 4 - Atende ao objetivo 3

Considere as seguintes tabelas:

Tabela Cliente

| Id_cliente | nome | endereço | telefone |
|------------|----------|---------------|------------|
| 1011 | Lucas P. | Rua X, 123 SP | 1124221312 |
| 1017 | Rosa M. | Rua Y, 111 RJ | 2122121111 |

Tabela Venda

| id_fatura | codigo_cliente | quantidade |
|-----------|----------------|------------|
| 12670 | 1011 | 25 |
| 90122 | 1017 | 31 |

Garantindo o sincronismo nas tabelas de Cliente e de Vendas, realize:

- a integridade referencial em ambas as tabelas.

b) a propagação de uma exclusão na tabela pai para a tabela filha.

Diagramação, inserir 10 linhas para a resposta.

Resposta comentada

a) Se desejamos estabelecer uma relação entre as tabelas de Cliente (tabela pai) e de Venda (tabela filha), é necessário obter um campo que seja comum a elas.

Observamos que o campo `id_cliente` da tabela Cliente e o campo `código_cliente` da tabela Venda têm algo em comum. Então, por esses campos poderemos estabelecer uma relação entre as tabelas. Por outro lado, na maioria das vezes o campo em comum deve ser a chave primária em alguma das tabelas. Supondo que a chave primária seja o campo `id_cliente` da tabela Cliente, criaremos a tabela chamada Cliente utilizando os comandos SQL:

```
CREATE TABLE Cliente(  
    id_cliente INT NOT NULL PRIMARY KEY,  
    nome VARCHAR(50) NOT NULL,  
    endereço VARCHAR(50) NOT NULL,  
    telefone CHAR(10) NOT NULL,  
);
```

Por outro lado, vamos considerar como chave primária o campo `id_fatura` na tabela Venda, e esse campo não pode ser nulo ==> `id_fatura` INT NOT NULL **PRIMARY KEY**,

Logo, vamos declarar a chave estrangeira no campo `código_cliente` na tabela Venda, garantindo assim a inclusão de uma venda caso exista a identificação do cliente correspondente ==> `código_cliente` INT **FOREIGN KEY**,



| id_fatura | código_cliente | quantidade |
|-----------|----------------|------------|
| 12670 | 1011 | 25 |
| 90122 | 1017 | 31 |
| 45132 | 211 | 43 |

Nas referências cruzadas utilizaremos os campos `código_cliente` na tabela de Venda

(tabela filha) e o campo `id_cliente` na tabela `Cliente` (tabela pai). A integridade referencial garante o sincronismo de valores entre a chave estrangeira `código_cliente` e a chave primária `id_cliente`. Finalmente, obtemos:

```
CREATE TABLE Venda
(
    id_fatura INT NOT NULL PRIMARY KEY,
    código_cliente INT NOT NULL,
    quantidade INT,
    código_cliente FOREIGN KEY,
    REFERENCES Cliente.id_cliente
);
```

Tabela Campo Chave primária – Tabela Cliente

b) A tabela pai é aquela que contém a chave primária e a tabela filha é a que possui a chave estrangeira. Então, em nosso caso, a tabela pai é a tabela `Cliente`, e a tabela filha é a tabela `Venda`. A propagação de uma exclusão na tabela pai para a tabela filha será dada pelos comandos ==> **ON DELETE CASCADE**.

Se desejamos incluir esse tipo de propagação de exclusão, devemos incluir o comando anterior no final da criação da tabela `Venda` da seguinte forma:

```
CREATE TABLE Venda
(
    id_fatura INT NOT NULL PRIMARY KEY,
    código_cliente INT NOT NULL,
    quantidade INT,
    código_cliente FOREIGN KEY,
    REFERENCES Cliente.id_cliente).
    ON DELETE CASCADE
```

A cláusula **ON DELETE CASCADE** indica que, se um registro da tabela pai (`Cliente`)

for apagado, o registro correspondente na tabela filha (Venda) também será apagado.

Fim da atividade

Início da atividade

Atividade 5 - Atende aos objetivos 2 e 3

A partir do modelo ER apresentado na Atividade 3, liste os comandos para criação das chaves estrangeiras:

- a) Criação de tabelas com índices e chaves estrangeiras;
- b) Criação de chaves estrangeiras supondo que tabelas e índices já existam.

Use a ferramenta MySQL para testar os comandos que você listou.

Diagramação, inserir 20 linhas para a resposta.

Resposta comentada

a) *Criação das tabelas com índices e chaves estrangeiras:*

Esquema relacional das tabelas

CLIENTE (CPF, Nome, Endereço, Telefone);

ORÇAMENTO (Número, Valor, Data, Descrição, CPF, Matricula);

SERVIÇO (NúmeroOrçamento, NotaFiscal, DataPagamento);

MECÂNICO (Matrícula, Nome, Telefone, Salário);

PEÇA (Código, Nome, Quantidade, Valor Unitário, Fabricante);

LISTA (CódigoPeça, NúmeroOrçamento);

EXECUTA (MatriculaMecânico, NúmeroServiço);

USA (CódigoPeça, NúmeroServiço);

Criação da tabela CLIENTE com índice:

CREATE TABLE CLIENTE (

 CPF VARCHAR(11) NOT NULL PRIMARY KEY,

 Nome VARCHAR(60) NOT NULL,

 Endereco VARCHAR(100) NOT NULL,

```
    Telefone VARCHAR(20) NOT NULL,  
    INDEX (CPF)  
);
```

Criação da tabela MECÂNICO com índice:

```
CREATE TABLE MECANICO (  
    Matricula INT NOT NULL PRIMARY KEY,  
    Nome VARCHAR(60) NOT NULL,  
    Telefone VARCHAR(20) NOT NULL,  
    Salario NUMERIC(9,2) NOT NULL,  
    INDEX (Matricula)  
);
```

Criação da tabela ORÇAMENTO com índice e chaves estrangeiras para as tabelas CLIENTE e MECÂNICO:

```
CREATE TABLE ORCAMENTO (  
    Numero INT NOT NULL PRIMARY KEY,  
    Valor NUMERIC(9,2) NOT NULL,  
    Data DATETIME NOT NULL,  
    Descricao VARCHAR(500),  
    CPF VARCHAR(11) NOT NULL,  
    Matricula INT,  
    INDEX (Numero)  
);
```

```
ALTER TABLE ORCAMENTO ADD CONSTRAINT FK_ORCAMENTO_CLIENTE  
FOREIGN KEY (CPF) REFERENCES CLIENTE(CPF);  
ALTER TABLE ORCAMENTO ADD CONSTRAINT FK_ORCAMENTO_MECANICO  
FOREIGN KEY (Matricula) REFERENCES MECANICO(Matricula);
```

Criação da tabela SERVIÇO com índice e chave estrangeira para a tabela ORCAMENTO:

```
CREATE TABLE SERVICO (
    NumeroOrcamento INT NOT NULL PRIMARY KEY,
    NotaFiscal VARCHAR(20),
    DataPagamento DATETIME,
    INDEX (NumeroOrcamento)
);
ALTER TABLE SERVICO ADD CONSTRAINT FK_SERVICO_ORCAMENTO
FOREIGN KEY (NumeroOrcamento) REFERENCES ORCAMENTO (Numero);
```

Criação da tabela PEÇA com índice:

```
CREATE TABLE PECA (
   Codigo INT NOT NULL PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Quantidade INT NOT NULL,
    ValorUnitario NUMERIC(9,2),
    Fabricante VARCHAR(100),
    INDEX (Codigo)
);
```

Criação da tabela LISTA com índice e chaves estrangeiras para as tabelas PECA e ORCAMENTO:

```
CREATE TABLE LISTA (
    CodigoPeca INT NOT NULL,
    NumeroOrcamento INT NOT NULL,
    PRIMARY KEY (CodigoPeca, NumeroOrcamento),
    INDEX (CodigoPeça, NumeroOrcamento)
);
```

```
ALTER TABLE LISTA ADD CONSTRAINT FK_LISTA_PECA FOREIGN KEY
(CodigoPeca) REFERENCES PECA (Codigo);
ALTER TABLE LISTA ADD CONSTRAINT FK_LISTA_ORCAMENTO FOREIGN KEY
(NumeroOrcamento) REFERENCES ORCAMENTO (Numero);
```

Criação da tabela EXECUTA com índice e chaves estrangeiras para as tabelas MECANICO e SERVICO:

```
CREATE TABLE EXECUTA (
    MatriculaMecanico INT NOT NULL,
    NumeroServico INT NOT NULL,
    PRIMARY KEY (MatriculaMecanico, NumeroServico),
    INDEX (MatriculaMecanico, NumeroServico)
);
```

```
ALTER TABLE EXECUTA ADD CONSTRAINT FK_EXECUTA_MECHANICO FOREIGN KEY (MatriculaMecanico) REFERENCES MECANICO (Matricula);
```

```
ALTER TABLE EXECUTA ADD CONSTRAINT FK_EXECUTA_SERVICO FOREIGN KEY (NumeroServico) REFERENCES SERVICO (NumeroOrcamento);
```

Criação da tabela USA com índice e chaves estrangeiras para as tabelas SERVICO e PECA:

```
CREATE TABLE USA (
    CodigoPeca INT NOT NULL,
    NumeroServico INT NOT NULL,
    PRIMARY KEY (CodigoPeca, NumeroServico),
    INDEX (CodigoPeca, NumeroServico)
);
```

```
ALTER TABLE USA ADD CONSTRAINT FK_USA_SERVICO FOREIGN KEY (NumeroServico) REFERENCES SERVICO (NumeroOrcamento);
```

```
ALTER TABLE USA ADD CONSTRAINT FK_USA_PECA FOREIGN KEY (CodigoPeca) REFERENCES PECA (CodigoPeca);
```

b) criação das chaves estrangeiras, supondo que tabelas e índices já existam:

Esquema relacional das tabelas

CLIENTE (CPF, Nome, Endereço, Telefone);

ORÇAMENTO (Número, Valor, Data, Descrição, CPF, Matricula);
SERVIÇO (NúmeroOrçamento, NotaFiscal, DataPagamento);
MECÂNICO (Matrícula, Nome, Telefone, Salário);
PEÇA (Código, Nome, Quantidade, Valor Unitário, Fabricante);
LISTA (CódigoPeça, NúmeroOrçamento);
EXECUTA (MatriculaMecânico, NúmeroServiço);
USA (CódigoPeça, NúmeroServiço);

Comandos para a criação das chaves estrangeiras:

- ALTER TABLE ORCAMENTO ADD CONSTRAINT FK_ORCAMENTO_CLIENTE FOREIGN KEY (CPF) REFERENCES CLIENTE(CPF);
- ALTER TABLE ORCAMENTO ADD CONSTRAINT FK_ORCAMENTO_MECHANICO FOREIGN KEY (Matricula) REFERENCES MECHANICO (Matricula);
- ALTER TABLE SERVICO ADD CONSTRAINT FK_SERVICO_ORCAMENTO FOREIGN KEY (NumeroOrcamento) REFERENCES ORCAMENTO (Numero);
- ALTER TABLE LISTA ADD CONSTRAINT FK_LISTA_PECA FOREIGN KEY (CodigoPeca) REFERENCES PECA (Codigo);
- ALTER TABLE LISTA ADD CONSTRAINT FK_LISTA_ORCAMENTO FOREIGN KEY (NumeroOrcamento) REFERENCES ORCAMENTO (Numero);
- ALTER TABLE EXECUTA ADD CONSTRAINT FK_EXECUTA_MECHANICO FOREIGN KEY (MatriculaMecanico) REFERENCES MECHANICO (Matricula);
- ALTER TABLE EXECUTA ADD CONSTRAINT FK_EXECUTA_SERVICO FOREIGN KEY (NumeroServico) REFERENCES SERVICO (NumeroOrcamento);
- ALTER TABLE USA ADD CONSTRAINT FK_USA_SERVICO FOREIGN KEY (NumeroServico) REFERENCES SERVICO (NumeroOrcamento);
- ALTER TABLE USA ADD CONSTRAINT FK_USA_PECA FOREIGN KEY

(CodigoPeca) REFERENCES PECA (CodigoPeca);

Fim da atividade

Agora vamos estudar e aplicar os comandos DML para que qualquer usuário interaja com os dados já armazenados no banco de dados. Vamos nessa!

Linguagem para manipulação de dados (DML)

A linguagem DML comprehende os comandos de manipulação e operação dos dados, ou seja, comandos de consulta e atualização dos dados no banco de dados, tais como:

- inserção de dados em uma tabela (**INSERT**);
- atualização de dados em uma tabela (**UPDATE**);
- remoção de dados de uma tabela (**DELETE**).

INSERT INTO - Utilizado para incluir registros nas tabelas do banco de dados. Sintaxe:

```
INSERT INTO nome_tabela  
(nome_campo1, nome_campo2,...)  
VALUES  
(novo_valor1, novo_valor2,...);
```

onde:

nome-tabela: representa o nome da tabela onde será incluído o registro.

nome-campo: representa o nome da(s) coluna(s) que terá conteúdo no momento da operação de inclusão.

VALUES: indica que, em seguida, serão colocados os valores dos campos do registro a ser incluído na tabela.

Exemplo: A seguinte tabela refere-se aos dados dos clientes:

| Código | Nome | Endereço | CEP | Cidade | UF |
|--------|---------------|------------|----------|----------------|----|
| 101 | João Silva | Rua S, 119 | 25650190 | São Paulo | SP |
| 102 | Lucas Hassen | Rua T, 21 | 25650191 | São Paulo | SP |
| 110 | Maria Pereira | Rua M, 13 | 25650198 | Rio de Janeiro | RJ |

Figura 8.9. Tabela Clientes.

Incluiremos os dados do cliente “Sebastian Oliveira” na tabela.

```
INSERT INTO Cliente (
    código_cliente, nome_cliente, endereço, cep, cidade, uf )
VALUES
(123, 'Sebastian Oliveira', 'Rua H, 231', '25650190', 'São Paulo'
'SP');
```

Obtemos em nossa tabela mais um registro de cliente.



| Código | Nome | Endereço | CEP | Cidade | UF |
|--------|--------------------|------------|----------|----------------|----|
| 101 | João Silva | Rua S, 119 | 25650190 | São Paulo | SP |
| 102 | Lucas Hassen | Rua T, 21 | 25650191 | São Paulo | SP |
| 110 | Maria Pereira | Rua M, 13 | 25650198 | Rio de Janeiro | RJ |
| 123 | Sebastian Oliveira | Rua H, 231 | 25650190 | São Paulo | SP |

UPDATE - Utilizado para alterar ou atualizar os dados contidos em uma tabela do banco de dados. Sintaxe:

```
UPDATE nome_tabela
SET nome_campo1 = novo_valor
WHERE
    condição;
```

onde:

nome_tabela: representa o nome da tabela cujo conteúdo será alterado.

nome_coluna: representa o nome da(s) coluna(s) que terá(ão) seu conteúdo alterado

com o novo valor especificado.

condição: representa a condição para a seleção dos registros que serão atualizados.

Essa seleção poderá resultar em um ou vários registros.

Exemplo: Considere a seguinte tabela Produto:

| Código Produto | Nome Produto | Nome Fornecedor | Valor Unitário |
|----------------|--------------|-----------------|----------------|
| 1235 | produto1 | J. S. | 13,6 |
| 3421 | produto2 | L. H | 12,5 |
| 1359 | produto3 | M. P. | 11,5 |



Figura 8.10. Tabela de Produto.

Tome por base os nomes dos campos: código_produto, nome_produto, nome_fornecedor e valor_unitário. Vamos atualizar o valor unitário do produto2:

UPDATE Produto

SET valor_produto = 15,7

WHERE

código_produto = 3121;

Nesse caso, o valor unitário de produto2, que era R\$12,5, passa a ser R\$15,7.

| Código Produto | Nome Produto | Nome Fornecedor | Valor Unitário |
|----------------|--------------|-----------------|----------------|
| 1235 | produto1 | J. S. | 13,6 |
| 3421 | produto2 | L. H | 15,7 |
| 1359 | produto3 | M. P. | 11,5 |



DELETE - Utilizado para excluir registros de uma tabela do banco de dados. Sintaxe:

```
DELETE FROM nome_tabela  
WHERE  
    condição;
```

onde:

nome-tabela: representa o nome da tabela cujos registros serão deletados.

condição: representa a condição para deletar os registros. Essa seleção poderá resultar em um ou vários registros.

Exemplo: Considere a seguinte tabela Vendedor:

| Código | Nome | Deptº | Data Admissão |
|--------|---------|-------|---------------|
| 1235 | Luis | 701 | 12/09/09 |
| 3421 | Carlos | 902 | 11/01/07 |
| 1359 | Liliane | 101 | 04/07/03 |

Figura 8.11. Tabela Vendedor.

Excluiremos os campos da vendedora Liliane:

DELETE FROM Vendedor

WHERE

código_vendedor = 1359;

Obtemos a seguinte tabela final:

| Código | Nome | Deptº | Data Admissão |
|--------|---------|-------|---------------|
| 1235 | Luis | 701 | 12/09/09 |
| 3421 | Carlos | 902 | 11/01/07 |
| 1359 | Liliane | 101 | 04/07/03 |

| Código | Nome | Deptº | Data Admissão |
|--------|--------|-------|---------------|
| 1235 | Luis | 701 | 12/09/09 |
| 3421 | Carlos | 902 | 11/01/07 |

Início da atividade

Atividade 6 - Atende ao objetivo 4

Considere a seguinte tabela Alunos:

| código | Nome | série | condição | situação |
|--------|-----------------------|-------|-----------|----------|
| 12312 | Lucas Prado Sifuentes | 8 | aprovado | AP |
| 12331 | Manoel Pontes | 7 | reprovado | RP |
| 12431 | Vanessa Hassem Miga | 7 | reprovado | RP |

Realize as seguintes manipulações:

- Certos pais desejam matricular seu filho na escola pela boa fama do colégio em exigência e disciplina; pelos documentos do novo aluno obtêm-se as seguintes informações: nome completo ==> Carlos Pereira Junior, série ==> 7^a, condição ==> aprovado (colégio anterior). Realize a inclusão desse aluno no banco de dados na tabela Alunos.
- Pelo motivo de aprovação do aluno Lucas, precisa-se que a tabela Alunos mostre a série atualizada, ou seja, se o aluno Lucas estava na oitava série, a tabela deverá mostrar a nona série.
- Os pais da aluna Vanessa decidem realizar transferência de colégio e a diretora necessita eliminar seus dados da tabela Alunos. Realize os comandos SQL apropriados para ajudar a diretora.

Diagramação: favor deixar 8 linhas para a resposta.

Resposta comentada

- Incluiremos os dados do aluno novo da seguinte maneira:

```
INSERT INTO Alunos (
    código, nome, serie, condição, situação)
VALUES
    (123432, 'Carlos Pereira Junior', "7", 'aprovado', 'AP');
```

- Vamos atualizar o campo série de oitava para nona série do aluno Lucas Prado da seguinte maneira:

UPDATE Alunos

SET serie = '9º'

WHERE

código = 12312;

c) A exclusão da aluna Vanessa será feita da seguinte maneira:

DELETE Alunos

WHERE código=12431;

Fim da atividade

Início da atividade

Atividade 7 - Atende ao objetivo 4

1. A partir do modelo ER apresentado na Atividade 3, liste os comandos SQL necessários para incluir os registros a seguir nas seguintes tabelas:

a) Cliente;

| CPF | Nome | Endereço | Telefone |
|-------------|-----------------|-----------------------------------|----------------|
| 83842848931 | Pedro Santos | Rua Maricá, 32 | 2471-8344 |
| 45475302122 | Zelda Zimmerman | Travessa Homilia, 72, 2º andar | (18) 4721-1561 |
| 39242928381 | Túlio Revelém | Praça dos Inválidos, 88 | 3118-6755 |

b) Peça;

| Código | Nome | Quantidade | Valor Unitário | Fabricante |
|--------|---------------|------------|----------------|------------|
| 11 | Filtro de ar | 50 | 140,00 | Fiat |
| 42 | Carburador | 120 | 270,00 | GM |
| 93 | Jogo de velas | 70 | 57,00 | Koala |

c) Orçamento

| Número | Valor | Data | Descrição | Cliente |
|--------|--------|------------|-------------------------------------|-------------|
| 12 | 270,00 | 18/11/2010 | Troca de carburador e jogo de velas | 83842848931 |
| 51 | 150,00 | 09/01/2011 | Substituição do filtro de ar | 45475302122 |

2. Liste os comandos SQL necessários para:

- a) Chegada de peças em estoque: atualize a quantidade de todas as peças acrescentando 100 unidades em cada peça;
- b) O endereço e o telefone da Zelda mudaram para: Rua do Rosário, 61 / (21) 3711-8236;
- c) O mecânico descobriu que seria necessário trocar o jogo de velas também para o orçamento número 51. Dessa forma, é necessário atualizar: o valor para R\$ 220,00 e a descrição, acrescentando nela o texto “e troca do jogo de velas”.
- d) O cliente Túlio desistiu de fazer os consertos, pode excluí-lo da tabela.

Use a ferramenta MySQL para testar os comandos que você listou.

Diagramação: favor deixar 10 linhas para a resposta.

Resposta comentada

1.

a)

```
INSERT INTO CLIENTE (CPF, Nome, Endereço, Telefone) VALUES
('83842848931','Pedro Santos','Rua Maricá, 32','2471-8344');
INSERT INTO CLIENTE (CPF, Nome, Endereço, Telefone) VALUES
('45475302122','Zelda Zimmerman','Travessa Homilia, 72, 2º andar','(18) 4721-1561');
INSERT INTO CLIENTE (CPF, Nome, Endereço, Telefone) VALUES
('39242928381','Túlio Revelém','Praça dos Inválidos, 88','3118-6755');
```

b)

```
INSERT INTO PECA (Código, NomeQuantidade, Valor Unitário, Fabricante) VALUES  
(11, 'Filtro de ar', 50, 140,00, 'Fiat');
```

```
INSERT INTO PECA (Código, NomeQuantidade, Valor Unitário, Fabricante) VALUES  
(42, 'Carburador', 120, 270,00, 'GM');
```

```
INSERT INTO PECA (Código, NomeQuantidade, Valor Unitário, Fabricante) VALUES  
(93, 'Jogo de velas', 70, 57,00, 'Koala');
```

c)

```
INSERT INTO ORCAMENTO (Numero, Valor, Data, Descrição, Cliente) VALUES (12,  
270,00, '18/11/2010', 'Troca de carburador e jogo de velas', '83842848931');
```

```
INSERT INTO ORCAMENTO (Numero, Valor, Data, Descrição, Cliente) VALUES (51,  
150,00, '09/01/2011', 'Substituição do filtro de ar', '45475302122');
```

2.

- a) UPDATE PECA SET Quantidade = Quantidade + 100;
- b) UPDATE CLIENTE SET Endereço = 'Rua do Rosário, 61', Telefone = '(21) 3711-
8236' WHERE CPF = '45475302122';
- c) UPDATE ORCAMENTO SET Valor = 220, Descricao = CONCAT (Descricao, 'e
troca do jogo de velas');
- d) DELETE FROM CLIENTE WHERE CPF = '39242928381';

Fim da atividade

Conclusão

Com o aumento das informações no banco de dados e um maior número de pessoas o acessando, podem surgir problemas nas operações de busca ou pesquisa, tornando-as mais lentas.

A utilização de índices é considerada uma ferramenta poderosa para um banco de dados, pois através deles conseguimos melhor desempenho para as consultas realizadas pelos diversos usuários. Com base nas informações proporcionadas nesta aula, você poderá tomar decisões importantes, como criar, alterar ou remover um índice, tendo em mente o equilíbrio entre as operações de leitura e escrita realizadas

num banco de dados.

Com a utilização dos comandos básicos (INSERT INTO, DELETE e UPDATE) você poderá resolver a maior parte dos problemas relacionados à manutenção e à extração de dados no banco de dados, assim como: inserção, atualização e remoção dos dados de uma tabela.

Resumo

Vamos rever os principais conceitos estudados nesta aula:

Índice: serve para prover um acesso rápido às linhas das tabelas. Por meio dele é possível unir uma ou mais colunas por onde o acesso é mais frequente.

Tipos de Índices:

Índice comum (**INDEX**):

- criam-se índices se os dados são muito utilizados em consultas e o tempo de resposta é insatisfatório.

Chave Primária (**PRIMARY KEY**):

- a chave primária é um tipo especial de índice que garante a singularidade: os valores nas colunas de chave primária são únicos ao longo de todas as linhas da tabela. Uma tabela pode ter apenas uma chave primária, e nenhuma das colunas na chave primária pode ser anulável;
- toda chave primária deve ser definida;
- MySQL requer que se especifique NOT NULL nas colunas que serão utilizadas como chaves primárias no momento da criação da tabela.

Índice único (**UNIQUE**):

- implica que não pode haver repetições na coluna ou combinação de colunas;
- é similar à restrição de chave primária, com duas exceções: um índice único pode conter colunas anuláveis e uma tabela pode ter qualquer número de índices únicos;
- são usados em chaves primárias.

Índice múltiplo:

- é um índice que agrupa “N” campos funcionando como um conjunto de “M”

índices diferentes, cada um formando um conjunto com os “x” primeiros campos indexados, para $x \leq N$;

- usado quando uma consulta deve ser executada utilizando “N” atributos.

A criação de índices para campos de uma tabela é o principal modo de otimização de acesso a dados fornecidos por MySQL.

Para especificar os campos indexados na criação da tabela, usa-se a palavra-chave **INDEX**, dentro da definição dos campos da tabela.

Para transformar em índice um campo já existente, usa-se a sintaxe de ALTER TABLE.

Para remover um índice, basta usar **DROP INDEX**.

O conceito de integridade referencial se refere à manutenção automática da consistência das referências cruzadas entre diferentes tabelas.

A integridade referencial:

- garante o sincronismo de valores entre a chave estrangeira (*foreign key*) tabela filha e a respectiva chave primária da tabela pai;
- propaga atualizações e exclusões efetuadas na tabela pai para a tabela filha em cascata.

A indicação explícita de chaves estrangeiras ajuda a manter a consistência em inserção de novas linhas, atualização de campos e exclusão de linhas.

A linguagem DML comprehende os comandos de manipulação e operação dos dados, ou seja, comandos de consulta e atualização dos dados no banco de dados, tais como:

- inserção de dados em uma tabela (**INSERT**);
- atualização de dados em uma tabela (**UPDATE**);
- remoção de dados de uma tabela (**DELETE**).

INSERT INTO - Utilizado para incluir registros nas tabelas do banco de dados.

UPDATE - Utilizado para alterar ou atualizar os dados contidos em uma tabela do banco de dados.

DELETE - Utilizado para excluir registros de uma tabela do banco de dados.

Informações sobre a próxima aula

Na próxima aula, você estudará os conceitos e a utilização da Álgebra Relacional, pois por meio dela poderemos realizar muitas operações de extração de dados num banco de dados.

Referências bibliográficas

- CARVALHO, C. R. 2006. *SQL - Guia prático*. 2^a ed. Rio de Janeiro: Brasport., 2006.
- DATE, C. J. *Introdução a sistemas de bancos de dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.
- ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.
- HEUSER, C. A. *Projeto de banco de dados*. 4^a ed. Porto Alegre: Bookman, 2009.
- SETZER, V. W.; SILVA, F. S. Corrêa da. *Bancos de dados*. São Paulo: Edgard Blucher, 2005.
- SILBERSCHATZ, A.; KORTH, H. *Sistema de banco de dados*. 3^a ed. São Paulo: Pearson Makron Books, 2008.

Disciplina: Modelando e implementando bancos de dados relacionais

Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 9

Consulta de dados em uma tabela - SQL

Meta

Apresentar a operação de seleção da álgebra relacional e o comando necessário para realizar essa operação no modelo físico de banco de dados.

Objetivos

Esperamos que, ao final desta aula, você seja capaz de:

1. Elaborar consultas utilizando operadores de seleção;
2. Aplicar consultas SQL de dados numa tabela usando os parâmetros:
 - WHERE
 - LIMIT
 - LIKE
 - ORDER BY

Pré-requisitos

Para ter bom aproveitamento desta aula, é importante que você tenha instalado e configurado a ferramenta MySQL conforme procedimento visto na Aula 6.

Introdução

Como os dados são extraídos em um banco de dados relacional? Agora você já sabe utilizar comandos SQL para criar tabelas e índices e incluir dados nessas tabelas. Mas é preciso também aprender como extrair ou obter do banco de dados as informações que desejamos. Afinal, o banco de dados foi criado exatamente para isso: armazenar os dados de forma segura e permitir a rápida recuperação das informações relevantes.

Para realizar consultas dentro das tabelas do banco de dados, você irá aprender um

novo comando, que é o SELECT. Com ele você poderá definir quais dados deseja obter e de onde quer obter esses dados ao executar as suas consultas.

Para recuperar no banco de dados apenas os pedidos de um determinado cliente “João da Silva”, por exemplo, você precisa dizer para o banco de dados que deseja obter os dados da tabela PEDIDOS e que o nome do cliente é igual a “João da Silva”. Aqui ensinaremos você a escrever as suas próprias consultas utilizando comandos SQL. Prepare-se para aprender a filtrar os dados das tabelas!

Fim da Introdução

Operações relacionais

Um banco de dados relacional é projetado para que os dados possam ser extraídos por operações relacionais e de conjuntos. Essas operações incluem:

- Seleção;
- Projeção;
- Junção.

Você conhecerá, nesta aula, a operação de Seleção (SELECT).

Comando SQL para operação de seleção: SELECT

Agora vamos estudar um dos comandos principais da linguagem SQL, o comando SELECT.

• Consultas de dados numa tabela

O comando SELECT é um comando de leitura utilizado para que o usuário possa efetuar consultas (*queries*), obtendo informações desejadas das tabelas do banco de dados. Para usar esse comando, você precisa indicar três coisas:

- 1) Quais dados você quer obter;
- 2) Onde estão esses dados;
- 3) Quais são os filtros usados para obter os dados.

Seu formato básico é:

```
SELECT nome_campo1, nome_campo2,...  
FROM nome_tabela
```

WHERE condição;

onde:

- SELECT: *nome_campo1, nome_campo2, ...*: retorna ou recupera os campos selecionados. O caractere * é utilizado para recuperar todos os campos da tabela.
- FROM *nome_tabela*: comando que indica o nome da tabela a ser consultada.
- WHERE condição: comando que expressa a condição (ou filtro) da consulta. Essa condição filtrará e permitirá a execução do comando apenas nos registros que satisfaçam a condição.

Exemplo: Para obter todos os dados dos pedidos do cliente “João da Silva”, podemos escrever o seguinte comando SQL:

```
SELECT *          (obtenha todos os dados)
FROM Pedidos     (da tabela pedidos)
WHERE NomeCliente = "João da Silva" (onde o nome do cliente seja João da Silva)
```

Dentro da cláusula WHERE você pode definir várias condições ou restrições para obter os dados utilizando os operadores de seleção.

Operadores de seleção

Os operadores de seleção são utilizados quando desejamos acrescentar condições especiais para obter registros específicos dentro do banco de dados.

Na matemática temos, por exemplo, a expressão $x > 5$. Ela restringe o valor de x a um determinado intervalo, que será apenas de números superiores a 5. Da mesma forma, você pode usar no banco de dados o operador de seleção maior que (>) para consultar dados específicos de uma tabela de funcionários.

Por exemplo, imagine que você queira filtrar os funcionários pelos valores do campo salário para obter da tabela apenas os funcionários cujo salário seja superior a R\$ 3.000,00. Nesse caso, basta acrescentar na sua consulta a seguinte restrição ou predicado: **salário > 3000**. Nesse caso, usamos o operador de seleção maior que (>).

Você também pode usar outros operadores para realizar comparações: = (igual a), ≠ (diferente de), < (menor que), <= (menor ou igual a), > (maior que) e >= (maior ou igual a).

O predicado dentro de uma consulta ao banco de dados é o conjunto de todas as condições utilizadas para filtrar os registros que desejamos consultar. Na prática, o predicado é tudo aquilo que vem depois da palavra WHERE, ou seja, todos os filtros que desejamos aplicar para obter os dados.

É possível também combinar o uso de vários operadores de seleção em conjunto, usando as palavras AND (e) e OR (ou) para fazer a ligação das condições ou filtros que forem utilizados.

Veja um exemplo para ficar mais fácil entender o que é um operador de seleção. Aplicamos os operadores de seleção para extrair um subconjunto de dados nas tabelas de Empregado e Departamento, utilizando o atributo chave Matricula.

Tabela Empregado

| Matricula | Nome | Cidade | Telefone |
|-----------|----------------|----------------|----------|
| 123456 | João Campos | Rio de Janeiro | 22113344 |
| 889977 | Pedro Gomes | São Paulo | 22351567 |
| 556633 | Maria do Carmo | Rio e Janeiro | 22451111 |

Favor corrigir para
Rio de Janeiro

Tabela Departamento

| Matricula | Departamento | Cargo | Data Entrada |
|-----------|--------------|-------------|--------------|
| 123456 | Pessoal | Funcionário | 12/04/04 |
| 889977 | Financeiro | Estagiário | 01/01/10 |
| 556633 | Pessoal | Chefe | 01/06/05 |

Figura 9.1: Tabelas Empregado e Departamento

a) Imagine que você precise fornecer as informações dos empregados cujas matrículas são maiores que 123456. Neste caso, podemos definir a condição usando o operador de seleção maior que (>):

Matricula > 123456.

A partir do atributo matricula, obtém-se uma descrição detalhada dos empregados com matricula maior que 123456 em ambas as tabelas. Obtivemos, assim, novas relações:

Tabela Empregado

| Matrícula | Nome | Cidade | Telefone |
|-----------|----------------|---------------|----------|
| 889977 | Pedro Gomes | São Paulo | 22351567 |
| 556633 | Maria do Carmo | Rio e Janeiro | 22451111 |

Favor corrigir cidade para Rio de Janeiro

Tabela Departamento

| Matrícula | Departamento | Cargo | Data Entrada |
|-----------|--------------|------------|--------------|
| 889977 | Financeiro | Estagiário | 01/01/10 |
| 556633 | Pessoal | Chefe | 01/06/05 |

Então os empregados cujas matrículas são maiores que 123456 são: Pedro Gomes, com matrícula 889977, e Maria do Carmo, com matrícula 556633. O empregado Pedro Gomes é da Cidade de São Paulo, com telefone 22351567, e trabalha no Departamento Financeiro como estagiário, com data de entrada em 01/10/10. Maria do Carmo é do Rio de Janeiro, com telefone 22451111, e trabalha no Departamento Pessoal como Chefe, com entrada em 01/06/05.

b) Agora encontre os nomes e os telefones de contato dos empregados que estão no Rio de Janeiro. A condição pode ser escrita da seguinte forma:

Cidade = "Rio de Janeiro".

Neste caso, precisamos só da Tabela Empregado para extrair os dados desejados.

Tabela Empregado

| Matrícula | Nome | Cidade | Telefone |
|-----------|----------------|----------------|----------|
| 123456 | João Campos | Rio de Janeiro | 22113344 |
| 556633 | Maria do Carmo | Rio e Janeiro | 22451111 |

Favor corrigir para Rio de Janeiro

Observe que existem dois empregados que estão no Rio de Janeiro. Seus nomes e telefones de contato são: João Campos, com telefone 22113344, e Maria do Carmo, com telefone 22451111.

c) Encontre os nomes dos empregados que pertençam ao Departamento Pessoal. O

operador de seleção será representado por:

Departamento = "Pessoal".

Pela tabela Departamento podemos conhecer as matrículas dos empregados que trabalham no Departamento Pessoal e, com essas matrículas, percorrer a Tabela Empregado para obter os nomes dos empregados. Os nomes dos empregados que pertencem ao Departamento Pessoal são: João Campos e Maria do Carmo.

Tabela Departamento

| Matricula | Departamento |
|-----------|--------------|
| 123456 | Pessoal |
| 556633 | Pessoal |

Tabela Empregado

| Matricula | Nome |
|-----------|----------------|
| 123456 | João Campos |
| 556633 | Maria do Carmo |

d) Forneça as informações dos empregados pertencentes ao Departamento Pessoal cujo registro Data de Entrada conste a partir de 01/01/05. Os operadores de seleção, usando o conetivo "e", serão representados por:

Departamento = "Pessoal" AND DataEntrada >= "01/01/05".

Sabemos que existem duas pessoas que trabalham no Departamento Pessoal.

Obtivemos, assim, as seguintes relações:

Tabela departamento

| Matricula | Departamento | Cargo | Data Entrada |
|-----------|--------------|-------|--------------|
| 556633 | Pessoal | Chefe | 01/06/05 |

Tabela Empregado

| Matricula | Nome | Cidade | Telefone |
|-----------|----------------|---------------|----------|
| 556633 | Maria do Carmo | Rio e Janeiro | 22451111 |

Favor corrigir para Rio de Janeiro

Logo, temos o caso do único empregado que satisfaz ambas as condições: é Maria do Carmo, com matrícula 556633, residente na cidade do Rio de Janeiro, possuindo cargo de Chefe e com telefone 22451111.

Finalmente, com esse e outros operadores de seleção, é possível realizar quaisquer

consultas utilizando álgebra relacional.

Início da atividade

Atividade 1 - Atende ao objetivo 1

Considere o seguinte modelo relacional:

Tabela Clientes

| Cód_cliente | Nome | Endereço | Cidade | Idade |
|-------------|-------|--------------------|--------|-------|
| 012 | Maria | Rua Bahia, 13 Lt14 | RJ | 23 |
| 022 | João | Rua Alagoas, 212 | SP | 45 |
| 036 | Marta | Rua Freitas, 712 | RS | 37 |

Tabela Contas

Tabela Cliente - Conta

| Nro_conta | Saldo – R\$ | Cód_cliente | Nro_conta |
|-----------|-------------|-------------|-----------|
| 783 | 1.100 | 012 | 531 |
| 531 | 1.850 | 022 | 783 |

Escreva as condições para executar as seguintes consultas relativas às Tabelas Clientes e Contas:

- Encontre os nomes dos clientes cujas idades sejam maiores que 25 anos.
- Encontre os nomes dos clientes cujas idades sejam maiores que 25 e cujos endereços sejam na cidade de SP.
- Forneça todas as informações dos clientes com saldos maiores e iguais a R\$1.500.

Diagramação, inserir um espaço de 10 linhas para cada item de resposta.

Resposta comentada

- Usando o operador de seleção maior que (>), as idades maiores que 25 anos podem ser representadas como: Idade > 25.

Nesse caso, precisamos só da Tabela Clientes para extrair os dados desejados. Obtivemos a seguinte nova relação:

| Cód_cliente | Nome | Endereço | Cidade | Idade |
|-------------|-------|------------------|--------|-------|
| 022 | João | Rua Alagoas, 212 | SP | 45 |
| 036 | Marta | Rua Freitas, 712 | RS | 37 |

Observamos que existem dois clientes cujas idades são maiores que 25 anos: João e Marta, cujos Cód_cliente são 022 e 036; seus endereços são Rua Alagoas, 212 – SP e Rua Freitas, 712 – RS, respectivamente.

- b) Usando o operador de seleção maior que ($>$) para representar as idades maiores que 25 anos e o operador de seleção igual a ($=$) para filtrar a cidade como "SP", obtemos: Idade > 25 AND Cidade = "SP".

Novamente, precisamos analisar apenas a Tabela Clientes. Dela obtivemos a seguinte nova relação:

| Cód_cliente | Nome | Endereço | Cidade | Idade |
|-------------|------|------------------|--------|-------|
| 022 | João | Rua Alagoas, 212 | SP | 45 |

- c) Utilizando o operador de seleção maior ou igual a (\geq), podemos representar o saldo maior e igual a R\$ 1.500 como: Saldo ≥ 1500

Nesse caso, precisamos consultar primeiro a Tabela Cliente-Conta para obter o relacionamento dos atributos Cód_cliente e Nro_conta. Pelo atributo Nro_conta, podemos extrair os saldos maiores e iguais a R\$ 1.500 na Tabela Contas. Obtivemos, assim, a seguinte relação:

| Nro_conta | Saldo |
|-----------|-------|
| 531 | 1.850 |
| 611 | 2.200 |

Observe que existem duas contas de clientes que possuem saldo maior ou igual a R\$ 1.500. Mas como desejamos fornecer todas as informações referentes aos clientes que possuem saldo maior ou igual a R\$ 1.500, então utilizamos a Tabela Clientes. Para poder acessar a Tabela Clientes, precisamos do atributo Cód_cliente, e esse atributo pode ser obtido da tabela de relacionamento Cliente_Conta.

| Cód_cliente | Nro_conta |
|-------------|-----------|
| 12 | 531 |
| 36 | 611 |

Pelo atributo Cód_cliente na Tabela Cliente-Conta, acessamos o Cód_cliente na Tabela Clientes, obtendo assim a seguinte nova relação:

| Cód_cliente | Nome | Endereço | Idade |
|-------------|-------|---------------------|-------|
| 12 | Maria | Rua Bahia, 13 RJ | 23 |
| 36 | Marta | Rua Freitas, 712 RS | 37 |

Finalmente, observamos que os dois clientes com saldo maior ou igual a R\$ 1.500 são: Maria, com endereço Rua Bahia, 13 – RJ, de 23 anos de idade e saldo igual a R\$ 1.850, e Marta, com endereço Rua Freitas, 712 – RS, 37 anos de idade e saldo igual a R\$ 2.200.

Fim da atividade

Os operadores de seleção podem ser divididos em operadores relacionais e operadores lógicos.

Operadores relacionais: uma forma de criar critérios é utilizando os operadores relacionais:

| Operadores relacionais | Significado |
|------------------------|------------------|
| = | Igual |
| > | maior que |
| < | menor que |
| >= | maior ou igual a |
| <= | menor ou igual a |

onde:

- ➔ = faz uma comparação de igualdade.
- ➔ > avalia se um termo possui valor maior do que outro.

- ➔ < avalia se um termo possui valor menor do que outro.
- ➔ >= avalia se um termo possui valor maior ou igual a outro.
- ➔ <= avalia se um termo possui valor menor ou igual ao de outro.

Operadores lógicos: em alguns casos, é preciso expressar condições utilizando os operadores lógicos.

| Operadores lógicos | Significado (utilizado para comparar expressões) |
|--------------------|---|
| And | e |
| Or | ou |
| Not | não |

onde:

- ➔ And significa "e" e permite juntar expressões.
- ➔ Or significa "ou" e permite que duas ou mais condições sejam testadas, esperando resultado verdadeiro de qualquer uma delas.
- ➔ Not é operador de negação.

Exemplos:

- A expressão *valor* = 200 avalia se a coluna *valor* de uma tabela é igual a 200.
- A expressão *valor* > 200 avalia se a coluna *valor* tem resultado maior do que 200.
- A expressão *valor* < 200 avalia se a coluna *valor* é menor do que 200.
- A expressão *valor* >= 200 avalia se a coluna *valor* é maior ou igual a 200.
- A expressão *valor* <= 200 avalia se a coluna *valor* é menor ou igual a 200.
- *valor* > 100 AND *valor* < 200. Aqui serão retornados apenas os registros que possuem valores maiores que 100 e menores que 200.
- *valor* = 100 OR *valor* = 200, serão retornados apenas os registros com valores iguais a 100 ou iguais a 200.
- Operador NOT inverte o valor de uma condição, ou seja, se a expressão *valor* = 200 retornar verdadeiro, a expressão NOT *valor* = 200 será falsa.

Veja agora como podemos escrever os comandos SQL usando um exemplo com uma tabela e diversos dados.

(1) Considere a tabela Funcionários com os seguintes nomes de campos: matricula, nome, endereço, data_admissão e salário.

| Matricula | Nome | Endereço | Data admissão | Salário |
|-----------|---------------|--------------------|---------------|---------|
| 1213 | Luis Oliveira | Rua E, 112 RJ | 03/12/10 | 623,55 |
| 2214 | Maria Silva | Rua Manoel, 321 RJ | 13/03/11 | 625,45 |
| 3219 | Pedro Campos | Rua Oliva, 12 RJ | 21/04/11 | 750,35 |
| 5674 | João Peres | Rua Fé, 43 RJ | 07/05/11 | 825,64 |

a) Deseja-se consultar na tabela Funcionários as informações matricula e nome de quem o salário seja maior do que R\$ 700.

Os comandos SQL correspondentes são:

SELECT matricula, nome ↙ campos da tabela a serem recuperados ou consultados.

FROM Funcionários ↙ nome da tabela

WHERE salário > 700; ↙ condição ou filtro que permitirá execução do comando apenas nos registros que satisfaçam os salários maiores que R\$700.

Com esses comandos, obtemos as seguintes informações:

| Matricula | Nome |
|-----------|--------------|
| 3219 | Pedro Campos |
| 5674 | João Peres |

(2) Veja a seguinte tabela Contas_a_Pagar:

| Código cliente | Data vencimento | Documento | Valor R\$ | Tipo |
|----------------|-----------------|-----------|-----------|------|
| 335 | 22/03/11 | 104 | 78 | A |
| 267 | 02/05/11 | 103 | 255 | B |

| | | | | |
|-----|----------|-----|-----|---|
| 543 | 03/03/11 | 101 | 432 | A |
| 121 | 11/04/11 | 102 | 543 | C |

- a) Agora é preciso recuperar os registros com todos os campos que a tabela Contas_a_Pagar possui cujo valor seja maior do que R\$ 300:

SELECT * recupera todos os campos da tabela.

FROM Contas_a_Pagar nome da tabela.

WHERE valor > 300; condição

Com esses comandos, obtemos as seguintes informações:

| Código Cliente | Data Vencimento | Documento | Valor | Forma |
|----------------|-----------------|-----------|-------|-------|
| 543 | 03/03/11 | 101 | 432 | A |
| 121 | 11/04/11 | 102 | 543 | C |

- b) Recuperar os registros com todos os campos que a tabela Contas_a_Pagar possui, cujos valores sejam menores ou iguais a R\$ 400:

SELECT *

FROM Contas_a_Pagar

WHERE valor <=400; valores menores ou iguais a R\$400.

Com esses comandos, obtemos a seguinte tabela:

| Código cliente | Data vencimento | Documento | Valor | Forma |
|----------------|-----------------|-----------|-------|-------|
| 335 | 22/03/11 | 104 | 78 | A |
| 267 | 02/05/11 | 103 | 255 | B |

- c) Recuperar os registros com todos os campos que a tabela Contas_a_Pagar possui cujos valores sejam entre R\$ 100 e R\$ 500:

SELECT *

FROM Contas_a_Pagar

WHERE valor >=100 **AND** valor <=500,00; valores entre R\$ 100 e R\$ 500.

Com esses comandos, obtemos a seguinte tabela:

| Código cliente | Data vencimento | Documento | Valor | Forma |
|----------------|-----------------|-----------|-------|-------|
| 267 | 02/05/11 | 103 | 255 | B |
| 543 | 03/03/11 | 101 | 432 | A |

- d) Recuperar os registros com todos os campos da tabela Contas_a_Pagar, com valores menores que R\$ 300 e valores maiores do que R\$ 500:

```
SELECT *
FROM Contas_a_Pagar
WHERE valor < 300 OR valor > 500,00;
```

↙ valores menores do que R\$300 ou
maiores do que R\$500.

Repare que, nesta situação, o conectivo OR (ou) serve para indicar que o caso em que qualquer uma das condições colocadas seja verdadeira é suficiente para recuperar o registro do banco de dados. Ou seja, ele irá obter o registro caso a primeira condição seja verdade ou caso a segunda condição seja verdade.

Com esses comandos, obtemos a seguinte tabela:

| Código cliente | Data vencimento | Documento | Valor | Forma |
|----------------|-----------------|-----------|-------|-------|
| 335 | 22/03/11 | 104 | 78 | A |
| 267 | 02/05/11 | 103 | 255 | B |
| 121 | 11/04/11 | 102 | 543 | C |

- e) Recuperar os registros com todos os campos da tabela Contas_a_Pagar com valores não maiores do que R\$ 100:

```
SELECT *
FROM Contas_a_Pagar
WHERE NOT valor > 100;
```

↙ valores não maiores do que R\$ 100.

| Código | Data | Documento | Valor | Forma |
|--------|------|-----------|-------|-------|
| | | | | |

| cliente | vencimento | | | |
|---------|------------|-----|----|---|
| 335 | 22/03/11 | 104 | 78 | A |

Início do Box de atenção

Nesse exemplo, o operador NOT aparece antes da expressão de comparação, o que inverte seu resultado; ou seja, quando as tuplas ou registros forem maiores que 100 a expressão retornará verdadeiro, mas o operador NOT a transformará em falso, não permitindo seu retorno. Quando a expressão for negativa, ou seja, quando o valor for menor ou igual a 100, o operador NOT a transformará em verdadeiro, permitindo, assim, o retorno das tuplas na consulta.

Fim do Box de atenção

Comando SQL - LIMIT

O parâmetro LIMIT é útil quando desejamos limitar a quantidade de registros retornados por uma consulta que foi executada. Isso costuma ocorrer quando existem muitos registros e o filtro de busca, por si só, não é suficiente para diminuir significativamente os dados recuperados. Para esses casos, é preciso limitar a quantidade de resultados ao fazer uma consulta no banco. O comando LIMIT permite que você especifique um numero máximo de linhas ou registros a serem exibidos e por qual linha ou registro você deseja começar a exibição.

Sintaxe geral: LIMIT valor;

LIMIT valor1, valor2;

onde:

valor: quantidade de valores retornados.

valor1, valor2 : valor inicial (valor1) e valor final (valor2) dos valores retornados.

Exemplo:

Considere a seguinte tabela Professor:

| Código | Nome | Área | Cidade-Estado | Telefone |
|--------|---------------|------------|---------------|----------|
| 12315 | Thais Solimar | Matemática | Caxias-RJ | 22 31 31 |

| | | | | |
|-------|---------------|-----------|--------------|----------|
| 23416 | Hugo Leal | Português | Madureira-RJ | 24 89 45 |
| 89012 | Sônia Miranda | Arte | Vassouras-RJ | 24 01 33 |
| 34189 | Teresa Santos | Biologia | Bela-SP | 31 45 67 |
| 21365 | João Pacheco | Física | Caxias/RJ | 22 31 44 |

a) Recuperar todos os campos da tabela, mostrando até os dois primeiros registros:

```
SELECT * FROM Professor LIMIT 2;
```

b) Recuperar as quantidades de registros que você quer por página, ou seja, suponha que você deseja recuperar os três registros a partir do registro 2:

```
SELECT * FROM Professor LIMIT 2 , 3;
```

c) O sistema conta a partir do 0 para recuperar os primeiros 5 registros:

```
SELECT * FROM Professor LIMIT 0 , 5;
```

Início da atividade

Atividade 2 - Atende ao objetivo 2

Considere uma tabela simples CR2IR (comprovante de rendimentos e de retenção de imposto de renda) com os seguintes nomes de campos: CPF, nome, total_rendimento, decimo_terceiro, ident_resp.

| CPF | Nome | Total de rendimentos | Décimo terceiro salário | Identidade responsável |
|--------------|----------------|----------------------|-------------------------|------------------------|
| 55.2838.7729 | Luiza Silva | R\$ 23.979,80 | 1.723,51 | Fundação DQ |
| 21.5212.2214 | Marta Oliveira | R\$ 7.280,60 | 525,78 | Instituto Pátria Nova |
| 13.1543.3219 | João Campos | R\$ 8.589,90 | 622,56 | Centro Cultural 123 |
| 56.7212.3674 | Lucas Barrera | R\$ 15.562,10 | 1.021,56 | Fundação AVIL |
| 06.1551.3414 | Tiago Cabral | R\$ 20.768,10 | 980,12 | Empresa XYZ |
| 32.2145.5671 | Matias Ruana | R\$ 25.979,10 | 2.111,78 | Instituto Brasil |

Aplique os comandos da linguagem SQL nos seguintes casos:

a) Deseja-se saber os nomes e total de rendimentos de todos os contribuintes que são obrigados fazer o imposto de renda (pelas informações, sabemos que está obrigado a declarar imposto de renda o contribuinte que no ano anterior recebeu rendimentos

- superiores a R\$ 15.764,28).
- b) Deseja-se conhecer todas as informações dos dois primeiros contribuintes que deverão declarar o imposto de renda.
 - c) Deseja-se saber os nomes das pessoas e as identidades dos responsáveis cujos valores do décimo terceiro salário variem entre R\$ 1.000 e R\$ 2.000.
 - d) Deseja-se saber os nomes das pessoas e as identidades dos responsáveis cujos valores do décimo terceiro salário sejam menores do que R\$ 1.000 ou maiores do que R\$ 2.000.
 - e) Recupere todas as identidades de todas as pessoas que são obrigadas a declarar imposto de renda.
 - f) Recupere os nomes e CPF das pessoas que não são obrigadas a declarar o imposto de renda.

Diagramação, inserir espaço referente a 15 linhas para a resposta.

Resposta comentada

- a) Para recuperar os nomes e total de rendimentos superiores a R\$ 15.764,28:

```
SELECT nome, total_rendimento  
FROM CR2IR  
WHERE total_rendimento >= 15.764,28,
```

obtendo a seguinte tabela:

| Nome | Total de Rendimentos |
|--------------|----------------------|
| Luiza Silva | R\$ 23.979,80 |
| Tiago Cabral | R\$ 20.768,10 |
| Matias Ruana | R\$ 25.979,10 |

- b) Recuperando todos os campos da tabela dos dois primeiros contribuintes que deverão declarar o imposto de renda.

```
SELECT *  
FROM CR2IR
```

WHERE total_rendimento >= 15.764,28

LIMIT 2,

obtendo a seguinte tabela:

| CPF | Nome | Total rendimentos | Décimo terceiro salário | Identidade responsável |
|--------------|--------------|-------------------|-------------------------|------------------------|
| 55.2838.7729 | Luisa Silva | R\$ 23.979,80 | 1.723,51 | Fundação DQ |
| 06.1551.3414 | Tiago Cabral | R\$ 20.768,10 | 980,12 | Empresa XYZ |

- c) Recuperando as informações dos nomes e as identidades responsáveis cujos valores do décimo terceiro salário variem entre R\$ 1.000 e R\$ 2.000.

SELECT nome, ident_resp

FROM CR2IR

WHERE décimo_terceiro > 1000 **AND** decimo_terceiro < 2000,

obtendo a seguinte tabela:

| Nome | Identidade Responsável |
|---------------|------------------------|
| Luisa Silva | Fundação DQ |
| Lucas Barrera | Fundação AVIL |

- d) Recuperando as informações dos nomes e as identidades responsáveis cujos valores do décimo terceiro salário sejam menores do que R\$ 1.000 ou maiores do que R\$ 2.000.

SELECT nome, ident_resp

FROM CR2IR

WHERE décimo_terceiro < 1000 **OR** decimo_terceiro > 2000;

obtendo a seguinte tabela:

| Nome | Identidade Responsável |
|----------------|------------------------|
| Marta Oliveira | Instituto Pátria Nova |
| João Campos | Centro Cultural 123 |
| Tiago Cabral | Empresa XYZ |
| Matias Ruana | Instituto Brasil |

e) Recupere todas as identidades responsáveis de todas as pessoas que são obrigadas a declarar imposto de renda.

```
SELECT ident_resp  
FROM CR2IR  
WHERE total_rendimento >= 15.764,28,
```

obtendo a seguinte tabela:

| |
|------------------------|
| Identidade responsável |
| Fundação DQ |
| Empresa XYZ |
| Instituto Brasil |

f) Recupere os nomes e CPF das pessoas que não são obrigadas a declarar o imposto de renda.

```
SELECT nome, CPF  
FROM CR2IR  
WHERE total_rendimento < 15.764,28;  
ou  
SELECT nome, CPF  
FROM CR2IR  
WHERE NOT total_rendimento >= 15.764,28;
```

obtendo a seguinte tabela:

| CPF | Nome |
|--------------|----------------|
| 21.5212.2214 | Marta Oliveira |
| 13.1543.3219 | João Campos |
| 56.7212.3674 | Lucas Barrera |

Fim da atividade

Comando SQL - LIKE

O que fazer nos casos em que se deseja realizar uma consulta e não tiver certeza

sobre o nome exato de um dado (produto ou nome de um aluno, por exemplo)?

Usaremos o comando LIKE. Esse comando recupera os nomes dos campos que terminam com algumas letras ou apenas as letras conhecidas. O procedimento a ser feito é o seguinte:

```
SELECT nome_campo1, nome_campo2,...  
FROM nome_tabela  
WHERE condição LIKE 'símbolo letrasconhecidas';
```

Onde o símbolo pode ser:

%: O símbolo % substitui qualquer número de caracteres desconhecidos. Então, LIKE '%letrasconhecidas' quer dizer que você está procurando todos os valores na coluna que terminam com letrasconhecidas.

_ : O símbolo sublinha sinaliza apenas um caractere desconhecido. Então, LIKE '_letrasconhecidas' quer dizer que você está procurando todos os valores com apenas um caractere antes das letrasconhecidas.

Exemplos:

1. “_ _ _ _ %” corresponde a qualquer texto com pelo menos quatro caracteres.
2. “Uni%” corresponde a qualquer texto que comece com “Uni” como, “universo”, “universal”, “universidade”.
3. "%a" corresponde a qualquer texto que comece com qualquer letra ou texto, mas que termine com a letra a.

Exemplo:

1. Considere a tabela Agenda:

| Nome | Sobrenome | e-mail | Cidade | Estado | telefone |
|--------|------------|------------------|------------|--------|----------|
| Carla | de Souza | clsou14@usml.com | Caxias | RJ | 213445 |
| Ana | Lima | lima90@usml.com | Bento | SC | 223541 |
| Camila | Oliveira | JJoo12@usml.com | Caxias | RJ | 275655 |
| Paola | dos Santos | sanpaty@hamh.com | Rio Bonito | SP | 243129 |

a) Gere a consulta mostrando todos os campos da tabela Agenda cujos nomes terminam com as letras **la**.

```
SELECT *
FROM Agenda
WHERE nome LIKE %la;
```

Obtemos os seguintes dados:

| Nome | Sobrenome | e-mail | Cidade | Estado | telefone |
|--------|------------|------------------|------------|--------|----------|
| Carla | de Souza | clsou14@usml.com | Caxias | RJ | 213445 |
| Camila | Oliveira | JJoo12@usml.com | Caxias | RJ | 275655 |
| Paola | dos Santos | sanpaty@hamh.com | Rio Bonito | SP | 243129 |

b) Gere a consulta utilizando todos os campos da tabela Agenda cujos nomes tenham apenas três letras e que terminem com **na**.

```
SELECT *
FROM Agenda
WHERE nome LIKE '_na';
```

Obtemos os seguintes dados:

| Nome | Sobrenome | e-mail | Cidade | Estado | telefone |
|------|-----------|-----------------|--------|--------|----------|
| Ana | Lima | lima90@usml.com | Bento | RJ | 223541 |

c) Gere a consulta utilizando todos os campos da tabela Agenda cujos nomes comecem com as letras **Ca**.

```
SELECT *
FROM Agenda
WHERE nome LIKE 'Ca%'
```

Obtemos os seguintes dados:

| Nome | Sobrenome | e-mail | Cidade | Estado | telefone |
|--------|-----------|------------------|--------|--------|----------|
| Carla | de Souza | clsou14@usml.com | Caxias | RJ | 213445 |
| Camila | Oliveira | JJoo12@usml.com | Caxias | RJ | 275655 |

d) Gere a consulta utilizando todos os campos da tabela Agenda, na qual os estados sejam RJ ou SC.

```
SELECT *
```

```
FROM Agenda
```

```
WHERE estado LIKE 'RJ' OR estado LIKE 'SC';
```

obtendo os seguintes dados:

| Nome | Sobrenome | e-mail | Cidade | Estado | telefone |
|--------|-----------|--|--------|--------|----------|
| Carla | de Souza | clsou14@usml.com | Caxias | RJ | 213445 |
| Ana | Lima | lima90@usml.com | Bento | SC | 223541 |
| Camila | Oliveira | JJoo12@usml.com | Caxias | RJ | 275655 |

Início da atividade

Atividade 3 - Atende ao objetivo 2

Considere uma tabela chamada Pacientes, com os campos cód, nome, endereço, data_insc, data_nasc, tipo_sangue, tipo_doença e situação:

| Código | Nome | Endereço | Data inscrição | Data nascimento | Tipo sangue | Doença | Situação |
|--------|--------------|--------------------|----------------|-----------------|-------------|----------|------------|
| 132112 | João Carlos | Rua EF, 11 RJ | 01/10/10 | 23/06/90 | O | Diabetes | Tratamento |
| 290113 | Ana Prado | Rua Tamí, 21 RJ | 09/03/09 | 11/05/94 | A | Dengue | Tratamento |
| 123343 | Luiz Silva | Rua Yam, 561 SP | 21/01/11 | 23/01/96 | AB | Dengue | Alta |
| 152718 | Maria Campos | Rua Nova, 33 RJ | 01/10/10 | 22/04/99 | O | Artrite | Tratamento |
| 135572 | Luz Oliveira | Rua Solaris, 81 AL | 30/03/99 | 25/07/93 | A | Dengue | Alta |
| 127888 | Luana Silva | Rua Yam, 561 SP | 12/07/01 | 30/03/94 | B | Dengue | Tratamento |

Aplique os comandos da linguagem SQL nos seguintes casos:

- Gere uma consulta mostrando todos os campos da tabela Pacientes de todos os nomes dos pacientes que começem com a letra **L**.
- Gere uma consulta mostrando todos os campos da tabela Pacientes de todas as doenças que começem com a letra **D** e cuja situação esteja em **Alta**.

c) Gere, a partir da tabela Pacientes, uma consulta na qual os pacientes tenham o tipo de sangue **A** ou combinação dele e morem no **RJ**. Mostrar os campos código, nome do paciente e tipo de doença.

Diagramação, inserir espaço de 10 linhas para a resposta.

Resposta comentada

a) Gerando a consulta de todos os pacientes cujos nomes começam com a letra L:

```
SELECT *
FROM Pacientes
WHERE nome LIKE 'L%';
```

Obtemos os seguintes dados:

| Código | Nome | Endereço | Data inscrição | Data nascimento | Tipo sangue | Doença | Situação |
|--------|--------------|--------------------|----------------|-----------------|-------------|--------|------------|
| 123343 | Luiz Silva | Rua Yam, 561 SP | 21/01/11 | 23/01/96 | AB | Dengue | Alta |
| 135572 | Luz Oliveira | Rua Solaris, 81 AL | 30/03/99 | 25/07/93 | A | Dengue | Alta |
| 127888 | Luana Silva | Rua Yam, 561 SP | 12/07/01 | 30/03/94 | B | Dengue | Tratamento |

b) Gerando a consulta de todas as doenças cujo nome começa com a letra **D** e cuja situação esteja em **Alta**.

```
SELECT *
FROM Pacientes
WHERE tipo_doença LIKE 'D%' AND situação='Alta';
```

Obtemos os seguintes dados:

| Código | Nome | Endereço | Data inscrição | Data nascimento | Tipo Sangue | Doença | Situação |
|--------|--------------|--------------------|----------------|-----------------|-------------|--------|----------|
| 123343 | Luiz Silva | Rua Yam, 561 SP | 21/01/11 | 23/01/96 | AB | Dengue | Alta |
| 135572 | Luz Oliveira | Rua Solaris, 81 AL | 30/03/99 | 25/07/93 | A | Dengue | Alta |

c) Gerando a consulta de todos pacientes que possuam tipo de sangue **A** ou combinação dele e morem no **RJ**, mostrando na consulta só os campos código, nome

do paciente e tipo de doença.

```
SELECT código, nome, tipo_doença  
FROM Pacientes  
WHERE tipo_sangue LIKE 'A%' AND endereço LIKE '%RJ' ;
```

Obtemos os seguintes dados:

| Código | Nome | Doença |
|--------|-----------|--------|
| 290113 | Ana Prado | Dengue |

Fim da atividade

Comando SQL – ORDER BY

Para realizar consultas ordenadas em uma tabela do banco de dados, basta utilizar o comando ORDER BY. Como isso é feito? Veja!

A consulta ordenada é realizada em função de um determinado campo. Este comando permite realizar consultas ordenadas, seja de forma ascendente ou descendente. Para isso, deve-se utilizar as opções ASC/DESC.

Box de atenção

Caso não seja definido se a ordem é ascendente ou descendente, o interpretador de comandos SQL assumirá o padrão, que é a ordem ascendente.

Fim do Box de atenção

O procedimento a ser feito é o seguinte:

```
SELECT nome_campo1, nome_campo2,...  
FROM nome_tabela  
ORDER BY nome_campo_especificado ASC/DESC;
```

Exemplo:

(1) Considere a seguinte tabela Produto:

| Código do produto | Código do fornecedor | Valor do produto |
|-------------------|----------------------|------------------|
| 6131 | 1213131 | 102,50 |
| 1101 | 3423121 | 321,70 |
| 2313 | 5242464 | 221,33 |

a) Seleccionaremos os produtos ordenados pelo código do Produto:

SELECT código_produto, código_fornecedor, valor_produto ← todos os campos da tabela

FROM Produto ← nome da tabela

ORDER BY código_produto; ← produtos ordenados pelo código do produto

ou

SELECT * ← todos os campos da tabela

FROM Produto

ORDER BY código_produto;

Obtemos os seguintes dados:

| Código do produto | Código do fornecedor | Valor do produto |
|-------------------|----------------------|------------------|
| 1101 | 3423121 | 321,70 |
| 2313 | 5242464 | 221,33 |
| 6131 | 1213131 | 102,50 |

(2) Suponha que temos a tabela Museu:

| Código | Nome da peça | Ano | Descrição |
|--------|-----------------|-----------|----------------------------------|
| 125643 | Livro 330 pág. | 900 d.C. | Escrito Rei Carlos II, Portugal |
| 561113 | Copo de ouro | 1100 d.C. | Utilizado Rainha Esther, Espanha |
| 431131 | Desenho animais | 550 a.C. | Feito pelo povo nômade, Europa |

a) Gerar uma consulta com todas as informações que a tabela Museu possua e que estejam ordenadas pela coluna de Descrição, em forma ascendente:

```

SELECT *           ← todos os campos da tabela
FROM Museu        ← nome da tabela
ORDER BY descrição ASC; ← coluna Descrição ordenada em forma
                        ascendente

```

Obtemos, assim, a seguinte consulta:

| Código | Nome da peça | Ano | Descrição |
|--------|-----------------|-----------|----------------------------------|
| 125643 | Livro 330 pág. | 900 d.C. | Escrito Rei Carlos II, Portugal |
| 431131 | Desenho animais | 550 a.C. | Feito pelo povo nômade, Europa |
| 561113 | Copo de ouro | 1100 d.C. | Utilizado Rainha Esther, Espanha |

b) Gerar uma consulta com todas as informações da tabela e que estejam ordenadas pelo nome da peça, em forma ascendente:

```

SELECT *           ← todos os campos da tabela
FROM Museu        ← nome da tabela
ORDER BY nome_peça DESC; ← peças ordenadas pelo nome em forma
                        descendente

```

Obtemos assim, a seguinte consulta:

| Código | Nome da peça | Ano | Descrição |
|--------|-----------------|-----------|----------------------------------|
| 561113 | Copo de ouro | 1100 d.C. | Utilizado Rainha Esther, Espanha |
| 431131 | Desenho animais | 550 a.C. | Feito pelo povo nômade, Europa |
| 125643 | Livro 330 pág. | 900 d.C. | Escrito Rei Carlos II, Portugal |

c) Gerar uma consulta ordenada pelo nome, em forma descendente, mostrando informações parciais, tais como nome, código e ano da peça:

```

SELECT código, nome_peça, ano   ← campos da tabela
FROM Museu                      ← nome da tabela
ORDER BY nome_peça DESC;         ← peças ordenadas pelo nome em forma
                                    descendente

```

Obtemos assim, a seguinte consulta:

| Código | Nome da peça | Ano |
|--------|-----------------|-----------|
| 125643 | Livro 330 pág. | 900 d.C. |
| 561113 | Copo de ouro | 1100 d.C. |
| 431131 | Desenho animais | 550 a.C. |

Início da atividade

Atividade 4 - Atende ao objetivo 2

Considere a tabela Aluno com os campos mat_aluno, nome, ano_nasc e curso:

| Matricula do aluno | Nome | Ano de nascimento | Curso |
|--------------------|---------------|-------------------|--------------------------|
| 126234 | Luis Oliveira | 12/06/97 | Tecnologia da Informação |
| 124389 | Maria Dolis | 01/03/98 | Ciências da Computação |
| 127467 | Ana Campos | 09/07/96 | Biotecnologia |
| 125612 | Ramon Costa | 02/05/90 | Sistemas de Informação |

Aplique os comandos da linguagem SQL nos seguintes casos:

- Gerar uma consulta com todas as informações da tabela, que estejam ordenadas pelo nome do aluno, em forma ascendente.
- Gerar uma consulta com as informações de matrícula, nome e curso, ordenadas pelo curso, em forma descendente.
- Gerar uma consulta ordenada pelo ano de nascimento, em forma ascendente, mostrando informações parciais, tais como nome, ano de nascimento e curso.

Diagramação, inserir 10 linhas para a resposta.

Resposta comentada

- Gerando a consulta com todas as informações da tabela e ordenadas pelo nome, ascendentemente:

```
SELECT *           ← todos os campos da tabela
FROM Aluno        ← nome da tabela
ORDER BY nome ASC; ← nomes ordenados em forma ascendentes
```

Obtemos assim, a seguinte consulta:

| Matricula do aluno | Nome | Ano de nascimento | Curso |
|--------------------|---------------|-------------------|--------------------------|
| 127467 | Ana Campos | 09/07/96 | Biotecnologia |
| 126234 | Luis Oliveira | 12/06/97 | Tecnologia de Informação |
| 124389 | Maria Dolis | 01/03/98 | Ciências da Computação |
| 125612 | Ramon Costa | 02/05/90 | Sistemas de Informação |

b) Gerando a consulta com as informações de matrícula, nome e curso, ordenadas pelo curso em forma descendente:

```
SELECT mat_aluno, nome, curso ← campos da tabela  
FROM Aluno ← nome da tabela  
ORDER BY curso DESC; ← curso ordenados em forma descendente
```

Obtemos assim, a seguinte consulta:

| Matricula do Aluno | Nome | Curso |
|--------------------|---------------|--------------------------|
| 126234 | Luis Oliveira | Tecnologia de Informação |
| 125612 | Ramon Costa | Sistemas de Informação |
| 124389 | Maria Dolis | Ciências da Computação |
| 127467 | Ana Campos | Biotecnologia |

c) Gerar uma consulta ordenada pelo ano de nascimento em forma ascendente, mostrando informações parciais, como nome, ano de nascimento e curso.

```
SELECT nome, ano_nasc, curso ← campos da tabela  
FROM Aluno ← nome da tabela  
ORDER BY ano_nasc ASC; ← ano ordenados em forma ascendente
```

Obtemos assim, a seguinte consulta:

| Nome | Ano de Nascimento | Curso |
|-------------|-------------------|------------------------|
| Ramon Costa | 02/05/90 | Sistemas de Informação |
| Ana Campos | 09/07/96 | Biotecnologia |

| | | |
|---------------|----------|--------------------------|
| Luis Oliveira | 12/06/97 | Tecnologia de Informação |
| Maria Dolis | 01/03/98 | Ciências da Computação |

Fim da atividade

Conclusão

Após estudar esta aula, você pode perceber que o poder de expressão da linguagem SQL é enorme. Ela possibilita aos usuários recuperar os dados contidos no banco com grande precisão, selecionando exatamente os registros que queremos mediante o uso das condições e dos operadores necessários.

Resumo

Para realizar consultas dentro das tabelas do banco de dados, você aprendeu um novo comando, que é o comando SELECT. Com ele você pode definir quais dados deseja obter e de onde quer obter esses dados ao executar as suas consultas.

Um banco de dados relacional é projetado para que os dados possam ser extraídos por operações relacionais e de conjuntos. Essas operações incluem: seleção, projeção e junção.

O operador de seleção seleciona tuplas que satisfazem um dado predicado. Em geral, podemos usar comparações do tipo `=, ≠, <, ≤, > e ≥`.

O comando SQL para operações de seleção é o SELECT. É um comando de leitura utilizado para que o usuário possa efetuar consultas (*queries*) obtendo subconjuntos das tabelas do banco de dados. Seu formato básico é:

```
SELECT nome_campo1, nome_campo2, ...
  FROM nome_tabela
 WHERE condição;
```

Em alguns casos você pode unificar duas ou mais condições utilizando os operadores lógicos AND e OR. Pode ainda inverter (negar) o efeito de uma condição utilizando o operador NOT.

O parâmetro LIMIT é praticamente obrigatório para sistemas que possuem grandes

bases de informações. Isso porque, nesses casos, é preciso limitar a quantidade de resultados ao fazer uma consulta no banco. Então o comando LIMIT permite que você especifique exatamente quantas linhas devem ser exibidas e por qual linha começar. O comando LIKE recupera os nomes dos campos que terminam com algumas letras ou apenas as letras conhecidas.

Para realizar consultas ordenadas em uma tabela do banco de dados, basta utilizar o comando ORDER BY. A consulta ordenada é realizada em função de um determinado campo. Esse comando permite realizar consultas ordenadas de forma ascendente ou descendente; para isso deve-se utilizar as opções ASC/DESC.

Informações sobre a próxima aula

Na próxima aula você verá como obter dados de diversas tabelas ou fontes de dados diferentes ao mesmo tempo, com o uso do operador de junção (JOIN).

Referências bibliográficas

CARVALHO, C. R. *SQL - guia prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.

DATE, C. J. *Introdução a sistemas de bancos de dados*. 8^a ed. Americana. Rio de Janeiro: Elsevier, 2003.

ELMASRI, R.; NAVATHE, S. *Sistemas de banco de dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.

HEUSER, C. A. *Projeto de banco de dados*. 4^a ed. Porto Alegre: Bookman, 2009.

SETZER, V. W.; SILVA, F. S. Corrêa da. *Bancos de dados*. São Paulo: Edgard Blucher, 2005.

SILBERSCHATZ, A.; KORTH, H. 2008. *Sistema de banco de dados*. 3^a ed. São Paulo: Pearson, 2008.

Aula 10

Consulta de dados em uma tabela – SQL (parte 2)

Meta

Apresentar comandos e funções importantes para o agrupamento de dados e a interligação de tabelas utilizando a linguagem padrão SQL.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Descrever as funções de agregação numéricas:
 - Função **COUNT**;
 - Função **SUM**;
 - Função **MAX/MIN**;
 - Cláusulas **GROUP BY** e **HAVING**.
2. Operações com conjuntos:
 - União (UNION);
 - Diferença (NOT IN);
 - Intersecção (INNER JOIN);
 - Produto cartesiano (CROSS JOIN).
3. Parâmetro de apelido **AS (Alias)**.

Pré-requisitos

Para se ter um bom aproveitamento desta aula, é importante que você recorde do comando **SELECT**, que foi ensinado na aula 9 e também os conceitos de chave primária e chave estrangeira, vistos na aula x.

[Introdução](#)

Na aula anterior você aprendeu como funciona o comando SELECT, utilizando operadores de seleção para recuperar dados específicos de uma tabela a partir da utilização de diversos filtros para a busca.

Além de buscar dados em uma tabela, você também pode agrupar esses dados utilizando algumas funções de agrupamento existentes na linguagem SQL. Por exemplo, se você utilizar a função de agrupamento MAX dentro do campo preço de uma tabela de produtos, ela irá trazer para você o valor máximo contido dentro do campo preço, ou seja, o produto com o maior preço existente na tabela de produtos. Estaremos vendo como funcionam esta e outras funções de agrupamento aqui nesta aula.

A linguagem SQL também permite extraímos dados de várias tabelas em conjunto com o comando SELECT. Suponha que você tenha pedidos de compra registrados numa tabela de pedidos e que você deseje buscar dados dos clientes que fizeram estes pedidos. Você pode interligar as tabelas utilizando o parâmetro de junção JOIN. Estaremos verificando com mais detalhes como esse parâmetro trabalha.

Fim da Introdução

A linguagem SQL possui algumas palavras-chave chamadas de funções. Vejamos:

I. Funções de Agregação numéricas em SQL

Funções são pedaços de códigos que realizam operações.

COUNT(*) : Exibe o número de linhas ou registros na tabela.

COUNT(nome_campo) : Exibe o número de linhas na coluna nome_campo. Se o valor for NULL ele não é contado.

SUM(nome_campo) : Soma os valores de uma coluna em todas as linhas ou registros.

AVG(nome_campo) : Média dos valores da coluna por todas as linhas.

MAX(nome_campo) : Valor máximo na coluna em todas as linhas.

MIN(nome_campo) : Valor mínimo na coluna em todas as linhas.

Exemplos: Considere a seguinte tabela Vendedor, com os campos código, nome, vendas e bairro:

| Código | Nome | Vendas R\$ | Bairro |
|--------|----------------|------------|------------|
| 110 | Márcio Pereira | 500 | Zona Sul |
| 114 | Jonas Baptista | 835 | Central |
| 210 | Maria Teixeira | 1 290 | Zona Norte |
| 212 | José Abreu | 800 | Zona Sul |
| 324 | Pedro Almeida | 1 000 | Central |
| 356 | Susan Neves | 900 | Zona Norte |
| 456 | Ricardo Gama | 650 | Zona Sul |

a) Quantas vendas foram feitas?

```
SELECT COUNT(*)
```

```
FROM Vendas;
```

| COUNT(*) |
|----------|
| 7 |

b) Qual a menor venda?

```
SELECT MIN(vendas)
```

```
FROM Vendedor;
```

| MIN(vendas) |
|-------------|
| 500 |

c) Qual a maior venda?

```
SELECT MAX(vendas)
```

```
FROM Vendedor;
```

| |
|-------------|
| MAX(vendas) |
| 1 290 |

d) Qual é o total de vendas na tabela Vendedor?

```
SELECT SUM(vendas)  
FROM Vendedor;
```

| |
|-------------|
| SUM(vendas) |
| 5 975 |

e) Qual é a venda total dos vendedores com código maior que 250?

```
SELECT SUM(vendas)  
FROM Vendedor  
WHERE código >250;
```

| |
|-------------|
| SUM(vendas) |
| 2 550 |

f) Quantos vendedores pertencem ao bairro Zona Norte?

```
SELECT COUNT(*)  
FROM Vendedor  
WHERE bairro = 'Zona Norte';
```

| |
|----------|
| COUNT(*) |
| 2 |

g) Quantos vendedores com vendas maior ou igual a R\$800 constam na tabela?

```
SELECT COUNT(*)
```

```
FROM Vendedor  
WHERE vendas > = 800;
```

| |
|----------|
| COUNT(*) |
| 5 |

h) Qual vendedor da Zona Sul tem maior venda?

```
SELECT MAX(vendas)  
FROM Vendedor  
WHERE bairro='Zona Sul';
```

| |
|-------------|
| MAX(vendas) |
| 800 |

j) Qual a venda total de todos os vendedores, exceto a da zona Central?

```
SELECT SUM(vendas) FROM Vendedor  
WHERE NOT(bairro = 'Central');
```

| |
|-------------|
| SUM(vendas) |
| 3 940 |

g) Qual é a média de todas vendas?

```
SELECT AVG(vendas)  
FROM Vendedor;
```

| |
|-------------|
| AVG(vendas) |
| 853,57 |

Início da atividade

Atividade 1 - Atende ao objetivo 1

(1) Considere a tabela AlunoNota com os campos mat_aluno, nome, nota e disciplina:

| Matricula do Aluno | Nome | Nota | Disciplina |
|--------------------|---------------|------|------------|
| 106234 | Luis Oliveira | 7,8 | Português |
| 104389 | Maria Dolis | 6,9 | Português |
| 107467 | Ana Campos | 8,1 | Biologia |
| 105612 | Ramon Costa | 7,2 | Português |
| 112311 | Rafael Castro | 6,7 | Matemática |
| 178222 | Rita Silva | 9,5 | Matemática |

Responda as seguintes perguntas utilizando os comandos de SQL:

- a) Quantos alunos existem na tabela?
- b) Qual a menor nota?
- c) Qual a maior nota?
- d) Quantos alunos pertencem à disciplina de Português?
- e) Qual aluno da disciplina de Português tem a maior nota?
- f) Qual é a média das notas da disciplina de Matemática?
- g) Qual é a média geral das notas?

Diagramação, inserir 7 linhas para a resposta.

Resposta Comentada

a) `SELECT COUNT(*)`

`FROM AlunoNota;`

| |
|----------|
| COUNT(*) |
| 6 |

b) SELECT MIN(nota)

```
FROM AlunoNota;
```

| |
|-----------|
| MIN(nota) |
| 6,7 |

c) SELECT MAX(nota)

```
FROM AlunoNota;
```

| |
|-----------|
| MAX(nota) |
| 9,5 |

d) SELECT COUNT(*)

```
FROM AlunoNota  
WHERE disciplina = 'Português';
```

| |
|----------|
| COUNT(*) |
| 3 |

e) SELECT MAX(nota)

```
FROM AlunoNota  
WHERE disciplina = 'Português';
```

| |
|-----------|
| MAX(nota) |
| 7,8 |

f) SELECT AVG(nota)

```
FROM AlunoNota  
WHERE disciplina = 'Matemática';
```

| |
|-----------|
| AVG(nota) |
| 8,1 |

g) SELECT AVG(nota)

```
FROM AlunoNota;
```

| |
|-----------|
| AVG(nota) |
| 7,2 |

Fim da Atividade

A instrução SELECT possui duas cláusulas poderosas: GROUP BY e HAVING.

GROUP BY: Indica os nomes dos campos a partir dos quais os valores serão agrupados e totalizados ou summarizados. Imagine, por exemplo, que tenhamos uma tabela que lista todas vendas realizadas numa loja. Poderemos descobrir o valor total das vendas por dia, se agruparmos o somatório (SUM) das vendas pelo campo de data da venda.

```
SELECT data, sum(valor)  
FROM venda  
GROUP BY data
```

A cláusula **HAVING** serve para acrescentar filtros ou condições, assim como a cláusula WHERE. A diferença, neste caso, é que o HAVING só é usado quando queremos realizar filtragem em cima dos campos envolvidos nos agrupamentos determinados pela cláusula GROUP BY. Estabelece condições para listar esses grupos. Dizemos que a cláusula HAVING está para a cláusula GROUP BY, assim como a cláusula WHERE está para o comando SELECT. As cláusulas WHERE e HAVING podem ser utilizadas ao mesmo tempo sem problemas.

Segundo o exemplo acima, para listarmos os dias que tiveram o total das vendas diárias superior a 3000 reais, podemos executar o seguinte comando SQL:

```
SELECT data, sum(valor)
FROM venda
GROUP BY data
HAVING sum(valor) > 3000
```

A sintaxe do comando SELECT com as cláusulas GROUP BY e HAVING é:

```
SELECT nome_campo1, nome_campo2, ...
FROM nome_tabela
WHERE condições
GROUP BY nome_campo_agrupamento
HAVING condição_agrupamento
```

Exemplos: Considere a seguinte tabela chamada Estágio

| Departamento | Nome Aluno | Quantidade horas |
|--------------|------------|------------------|
| TI | Lucas | 40 |
| TI | Sônia | 35 |
| TI | Beatriz | 38 |
| CC | Sônia | 25 |
| CC | Lucas | 31 |
| SI | Beatriz | 36 |
| SI | Sônia | 29 |

a) Desejamos saber o somatório de horas totais da tabela Estágio, então utilizaremos SUM(qtdHoras) junto com o comando SELECT:

```
SELECT SUM (qtdeHoras)
FROM Estágio;
```

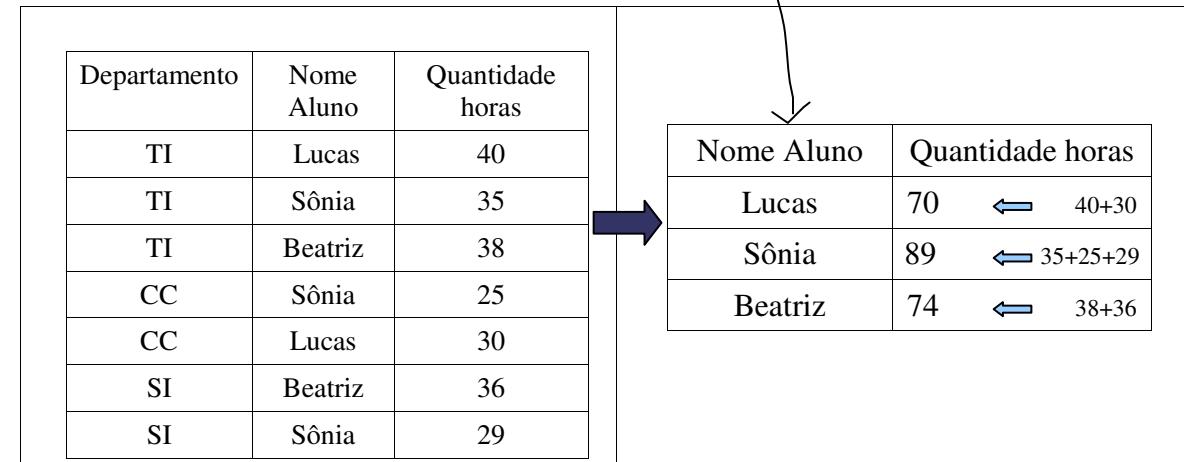
Nosso resultado será:

| Quantidade horas |
|------------------|
| 234 |

b) Desejamos saber o somatório de horas totais por aluno:

```
SELECT nome_aluno, SUM (qtdeHoras)
FROM Estágio
GROUP BY nome_aluno;
```

Nosso resultado será:



c) E se sobre os grupos obtidos na consulta anterior, ainda se desejar fazer um filtro através de uma condição como total de horas superior a 70 horas:

```
SELECT nome_aluno, SUM (qtdeHoras)
FROM Estágio
GROUP BY nome_aluno HAVING SUM (qtdeHoras) > 70;
```

Nosso resultado será:

| Nome Aluno | Quantidade horas |
|------------|------------------|
| Sônia | 89 |
| Beatriz | 74 |

d) E se sobre os grupos obtidos na consulta anterior, ainda se desejar realizar mais um filtro, isto é, ordenar pelos nomes dos alunos em forma ascendente:

```
SELECT nome_aluno, SUM (qtdeHoras)
FROM Estágio
```

```

GROUP BY nome_aluno HAVING SUM(qtdeHoras) > 70
ORDER BY nome_aluno ASC;

```

Nosso resultado será:

| Nome Aluno | Quantidade horas |
|------------|------------------|
| Beatriz | 74 |
| Sônia | 89 |

Início da atividade

Atividade 2 - Atende ao objetivo 1

(1) Considere a tabela Cliente com os campos código, nome_cliente, data_compra e valor:

| Código | Nome | Data Compra | Valor |
|--------|---------------|-------------|--------|
| 126234 | Luis Oliveira | 12/06/10 | 121,65 |
| 124389 | Maria Dolis | 01/03/10 | 32,77 |
| 127467 | Ana Campos | 09/07/10 | 89,55 |
| 125612 | Ramon Costa | 02/05/10 | 230 |
| 124389 | Maria Dolis | 22/07/10 | 56,3 |
| 126234 | Luis Oliveira | 03/09/10 | 98,22 |
| 127467 | Ana Campos | 09/11/10 | 167,54 |

- a) Desejamos saber os valores de compra por cliente.
- b) Desejamos saber as médias dos valores de compra por cliente.
- c) Agora, nós precisamos do resultado anterior, ordenar pelos códigos dos clientes.

Diagramação, inserir 7 linhas para a resposta.

Resposta Comentada

a) Agrupamos todos os nomes para cada cliente e o total de compras por grupo de cliente:

```

SELECT código, nome_cliente, SUM(valor)
FROM Cliente
GROUP BY nome_cliente;

```

Nosso resultado será:

| Código | Nome | Valor |
|--------|------|-------|
| | | |
| | | |

| | | |
|--------|---------------|--------|
| 126234 | Luis Oliveira | 219,87 |
| 124389 | Maria Dolis | 89,07 |
| 127467 | Ana Campos | 257,09 |
| 125612 | Ramon Costa | 230 |

b) Agrupamos todos os nomes para cada cliente, mas desta vez estamos obtendo a média dos valores de compra do cliente:

```
SELECT código, nome_cliente, AVG(valor)
FROM Cliente
GROUP BY nome_cliente;
```

Nosso resultado será:

| Código | Nome | Valor |
|--------|---------------|--------|
| 126234 | Luis Oliveira | 109,95 |
| 124389 | Maria Dolis | 44,54 |
| 127467 | Ana Campos | 128,55 |
| 125612 | Ramon Costa | 230 |

c) Ordenando pelos códigos dos clientes do resultado anterior:

```
SELECT código, nome_cliente, AVG(valor)
FROM Cliente
GROUP BY nome_cliente
ORDER BY código;
```

Nosso resultado será:

| Código | Nome | Valor |
|--------|---------------|--------|
| 124389 | Maria Dolis | 44,54 |
| 125612 | Ramon Costa | 230 |
| 126234 | Luis Oliveira | 109,95 |
| 127467 | Ana Campos | 128,55 |

E se você precisar recuperar as informações em mais de uma tabela? Neste caso, basta utilizar o comando JOIN, mas este comando é utilizado dependendo dos relacionamentos entre as tabelas. Vamos ver como se utilizar esse comando!

[Fim da atividade](#)

II. Operações de Conjunto

São operações e elas são provenientes da teoria de conjuntos da matemática. Eles atuam sobre um ou mais conjuntos de linhas (tabelas) para produzir um novo conjunto de linhas (tabela resultante). As operações de conjunto são:

- União;
- Diferença, Subtração;
- Interseção;
- Produto Cartesiano.

Vamos examinar alguns exemplos utilizando a Tabela de Produto 1 e a Tabela de produto 2.

| Nome Produto | Preço Unitário (R\$) |
|------------------|----------------------|
| Caneta Azul Pic | 1,2 |
| Régua 30cm | 0,9 |
| Caderno 50folhas | 1,5 |

| Nome Produto | Preço Unitário (R\$) |
|---------------------|----------------------|
| Caneta Vermelha Pic | 1,2 |
| Régua 30cm | 0,9 |
| Borracha branca | 0,8 |

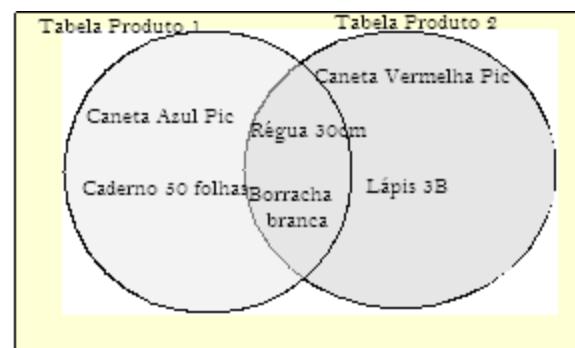
Figura 4.4 Tabelas de Produto 1 e Tabela de produto 2

União

A operação de união permite extrair todos os produtos inclusos na Tabela de Produto 1 e na Tabela de Produto 2. O resultado é o seguinte:

| Nome Produto | Preço Unitário (R\$) |
|---------------------|----------------------|
| Caneta Azul Pic | 1,2 |
| Régua 30cm | 0,9 |
| Caderno 50folhas | 1,5 |
| Borracha branca | 0,8 |
| Caneta Vermelha Pic | 1,2 |
| Lápis 3B | 1,1 |

A figura a seguir mostra como ficam os dados das duas tabelas depois que a operação



foi executada.

Sintaxe do comando SQL para executar esta operação:

```
SELECT campo1, campo2, ..., campoN FROM tabela1  
UNION  
SELECT campo1, campo2, ..., campoN FROM tabela2
```

Exemplo de uso:

Vamos supor que você queira unificar os produtos das duas tabelas de produtos descritas acima. O comando neste caso é:

```
SELECT Nome, PrecoUnitario FROM PRODUTO1  
UNION  
SELECT Nome, PrecoUnitario FROM PRODUTO2
```

Observações:

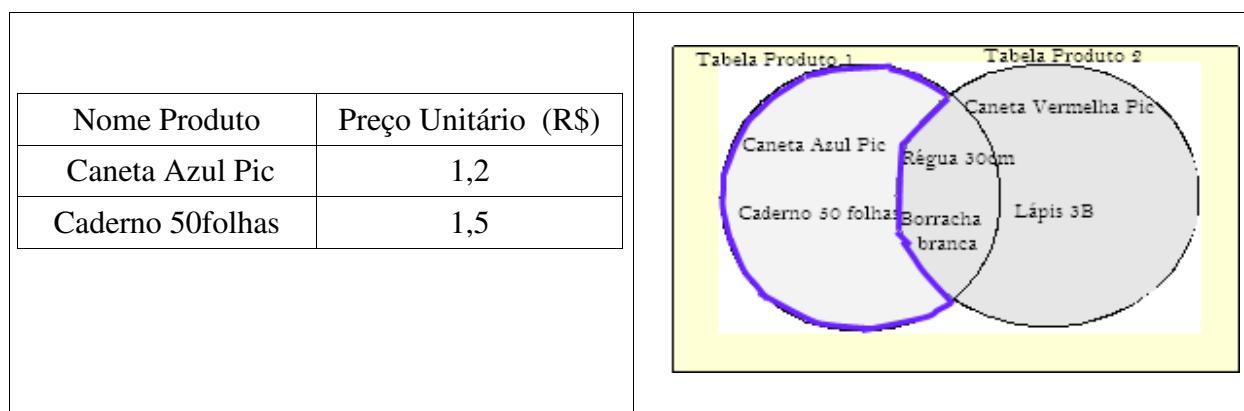
Veja que os campos selecionados nas duas tabelas precisam ser do mesmo tipo para que o comando UNION funcione. No exemplo acima, os campos Nome e PrecoUnitario são iguais nas duas tabelas de produtos.

Repare que ao utilizar a cláusula UNION, os registros ou linhas não aparecerão duplicados apesar de estarem iguais nas duas tabelas de produtos. Caso você deseje exibir também todos os registros duplicados (fazer os itens régua e borracha aparecerem duas vezes, por exemplo) você deve utilizar a cláusula UNION ALL ao

invés de UNION para unir os conteúdos das tabelas.

Diferença

É uma operação que extrai linhas de apenas uma das tabelas, ou seja, a diferença entre a Tabela de Produto 1 e a Tabela de Produto 2, é a extração de todos os produtos da primeira tabela que não estão incluídos na segunda tabela:



Sintaxe do comando SQL para executar esta operação:

```
SELECT campo1, campo2, ..., campoN FROM tabela1  
WHERE ( campo1, campo2, ..., campoN) NOT IN  
(select campo1, campo2, ..., campoN FROM tabela2)
```

Exemplo de uso:

Para executar a operação de diferença de conjuntos nas tabelas de produtos apresentadas acima, execute o seguinte comando SQL:

```
select Nome, PrecoUnitario from produto  
WHERE ( Nome, PrecoUnitario) NOT IN  
(select Nome, PrecoUnitario from produto2)
```

Início da atividade online

Atividade 3 - Atende ao objetivo 2

Tio Olegário possui uma fazenda em Guapimirim. Ele fez uma relação de alimentos

que são vendidos nos armazéns mais próximos de sua fazenda e também listou os animais de sua fazenda e que tipo de alimento consomem.

Tabela ALIMENTO_VENDIDO

| Código | Nome |
|--------|----------|
| 1 | Grãos |
| 2 | Verduras |
| 3 | Milho |
| 4 | Carne |
| 5 | Peixe |
| | |
| | |
| | |
| | |

Tabela ANIMAL_FAZENDA

| Código | Nome | Alimento |
|--------|----------|----------|
| 1 | Cachorro | Carne |
| 2 | Cavalo | Grama |
| 3 | Boi | Grama |
| 4 | Galinha | Milho |
| 5 | Pombo | Milho |
| 6 | Porco | Grãos |
| 7 | Coelho | Cenoura |
| 8 | Gato | Peixe |
| | | |

O Tio Olegário precisa gerar as seguintes listagens (Escreva o comando SQL necessário para gerar a listagem):

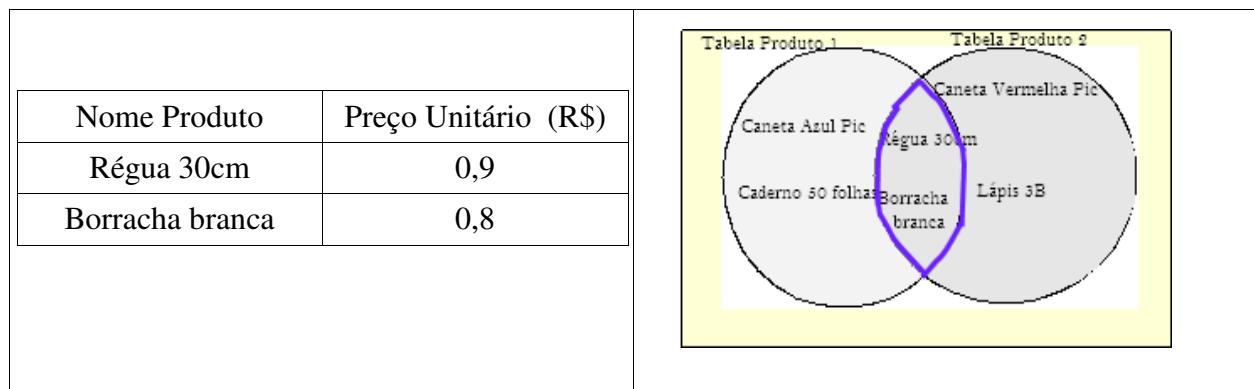
- Listagem completa de alimentos, incluindo os nomes de todos os alimentos vendidos e consumidos na fazenda;
- Listagem dos alimentos vendidos mas que não são consumidos na fazenda.

Fim da atividade online

Interseção (ou junção)

É a operação mais freqüente utilizada na linguagem SQL. Extrai as linhas que estão inclusas em ambas as Tabelas, seguindo um ou mais critérios de interseção ou junção.

Veja a seguir o resultado da intersecção das tabelas de produtos 1 e 2:



Sintaxe do comando SQL para executar esta operação:

```
SELECT campo1, campo2, ..., campoN
FROM tabela1
INNER JOIN tabela2 ON tabela1.campoX = tabela2.campoX
```

Exemplo de uso:

Vamos digitar o seguinte comando SQL para obter os nomes e os preços dos produtos que estão presentes nas duas tabelas de produtos, realizando a junção das tabelas através do campo nome do produto:

```
SELECT produto1.codigo, produto1.nome
FROM produto1
INNER JOIN produto2 ON produto1.nome = produto2.nome
```

Repare que este comando é utilizado com freqüência para juntar dados de tabelas relacionadas a partir de um campo chave definido. Veja um exemplo disso:

A tabela **Aluno** tem o campo Código Unidade que se relaciona com o campo Identificação Unidade na tabela **Unidade**, sendo obrigatório os dois campos terem o mesmo tipo.

Tabela Aluno

| Código Unidade | Nome | Endereço | |
|----------------|------------------|----------------------------|------|
| 110101 | Maria Silva | Rua Bahia Lt13 Qd 50 DQ-RJ | |
| 212201 | Rita Mar Costa | Rua Rosa 11 NI-RJ | |
| 212201 | João Campos | Rua F 24 DQ-RJ | |
| 431002 | Ana <u>Nagir</u> | Rua Passagem 673 DQ-RJ | |

Tabela Unidade

| Identificação Unidade | Nome Unidade | Endereço | |
|-----------------------|--------------|----------------------------|------|
| 110101 | Bom Estudo | Rua Getúlio 132 RJ | |
| 212201 | Santa Fé | Rua Floriano 342 DQ-RJ | |
| 431002 | Terra Nova | Rua <u>Cipro</u> 750 NI-RJ | |

Como podemos fazer para obter ao mesmo tempo o nome dos alunos e o nome das unidades onde eles estudam?

Nesse caso, para obtermos os dados de ambas as tabelas, o comando **INNER JOIN** pode ser usado da seguinte forma:

```
SELECT ALUNO.Nome, UNIDADE.NomeUnidade
FROM ALUNO
INNER JOIN UNIDADE
ON ALUNO.CodigoUnidade = UNIDADE.IdentificacaoUnidade
ORDER BY ALUNO.Nome
```

Onde:

- **SELECT ALUNO.Nome, UNIDADE.NomeUnidade:** Seleciona o campo Nome da tabela **ALUNO** e campo NomeUnidade da tabela **UNIDADE**.
- **FROM ALUNO:** A origem é a tabela **ALUNO**. Isso significa que a tabela **ALUNO** é o ponto de partida para a realização da consulta. Você poderia colocar a tabela **UNIDADE** como origem se preferisse, mas como a tabela ALUNO possui mais registros, é melhor colocar ela como ponto de partida para tornar a consulta mais ágil.
- **INNER JOIN UNIDADE:** A tabela **ALUNO** será ligada a tabela **UNIDADE**.
- **ON ALUNO.CodigoUnidade = UNIDADE.CodigoUnidade:** Faz-se aqui o relacionamento das tabelas.
- **ORDER BY ALUNO.Nome:** A pesquisa dos nomes dos alunos e suas

unidades será exibida ordenada alfabeticamente pelo nome de aluno.

Como resultado da operação de junção e consulta SQL, teremos a seguinte tabela:

| Nome | Nome Unidade |
|----------------|-----------------|
| Ana Nagir | Terra Nova |
| João Campos | Santa Fé |
| Maria Silva | Bom Estudo |
| Rita Mar Costa | Santa Fé |

Produto Cartesiano

É um método que combina todas as linhas nas duas tabelas.

Veja as seguintes tabelas:

| Tabela Funcionário | | | Tabela Departamento | |
|--------------------|-------|---------------------|---------------------|------------|
| Código | Nome | Endereço | CódDept | Nome Dept. |
| 1123 | Lucas | Rua Maya 123 BL. RJ | 12 | Pessoal |
| 1221 | Maria | Rua A Lt 1, VN. SP | 32 | Financeiro |

Figura. 4.5 Exemplo de chaves estrangeiras para a tabela de funcionários.

A operação de produto cartesiano combina todas as linhas nas duas tabelas. Neste exemplo obteremos $3 \times 2 = 6$ linhas da seguinte forma:

| Código | Nome | Endereço | CódDept | Nome Dept. |
|--------|-------|---------------------|---------|------------|
| 1123 | Lucas | Rua Maya 123 BL. RJ | 12 | Pessoal |
| 1123 | Lucas | Rua Maya 123 BL. RJ | 32 | Financeiro |
| 1221 | Maria | Rua A Lt 1, VN. SP | 12 | Pessoal |
| 1221 | Maria | Rua A Lt 1, VN. SP | 32 | Financeiro |
| 1332 | Pedro | Rua Bahia 12 Qt. RJ | 12 | Pessoal |
| 1332 | Pedro | Rua Bahia 12 Qt. RJ | 32 | Financeiro |

Sintaxe do comando SQL para executar esta operação:

SELECT campo1, campo2, ..., campoN

```
FROM tabela1  
CROSS JOIN tabela2
```

Exemplo de uso:

Veja como podemos escrever o comando SQL para realizar a operação de produto cartesiano entre as tabelas FUNCIONARIO e DEPARTAMENTO que foram listadas, exibindo todos os campos de ambas as tabelas:

```
SELECT FUNCIONARIO.* , DEPARTAMENTO.*  
FROM FUNCIONARIO  
CROSS JOIN DEPARTAMENTO
```

Início da atividade online

Atividade 4 - Atende aos objetivos 1 e 2

A locadora “Alugue Agora” possui um acervo de filmes e lançamentos à disposição de seus clientes. Recentemente, ela está montando uma classificação dos filmes oferecidos por gênero (drama, terror, ficção, ...). Seguem as listagens dos gêneros dos filmes e dos lançamentos oferecidos pela locadora.

Tabela GENERO_FILME

| Código | Nome |
|--------|----------|
| 1 | Drama |
| 2 | Aventura |
| 3 | Ação |
| 4 | Terror |
| 5 | Ficção |
| 6 | Romance |
| 7 | Comédia |
| | |

Tabela LANCAMENTO

| Código | Nome | Cód. Gênero |
|--------|----------------------------|-------------|
| 1 | Fúria sobre rodas | 3 |
| 2 | Tron | 5 |
| 3 | Avatar | 5 |
| 4 | Se beber, não case | 7 |
| 5 | Thor | 3 |
| 6 | Incêncios | 1 |
| 7 | Reencontrando a Felicidade | 1 |
| 8 | Arrasando | 6 |

| | |
|--|--|
| | |
| | |

| | | |
|--|----------|--|
| | Corações | |
| | | |

Tabela FILIAL

| Código | Nome | Clientes |
|--------|---------------|----------|
| 1 | Madureira | 3452 |
| 2 | Centro | 27281 |
| 3 | Vargem Grande | 1230 |
| 4 | Tijuca | 1327 |
| 5 | Campo Grande | 4238 |
| 6 | Niterói | 6360 |
| 7 | Paracambi | 13518 |
| | | |
| | | |

A locadora precisa gerar as seguintes listagens (Escreva o comando SQL necessário para gerar a listagem):

- a) Listagem dos nomes dos filmes incluindo os nomes dos gêneros aos quais eles estão associados;
- b) Listagem de nomes dos lançamentos x nomes das filiais da locadora, pois a locadora deseja enviar originais de todos os lançamentos listados para cada uma das filiais listadas na tabela FILIAL;
- c) Listagem do total de clientes da locadora “Alugue Agora”.

Fim da atividade online

Parâmetro de Apelido AS (Alias)

Quando você utiliza comandos SQL envolvendo várias tabelas, você pode dispor de um recurso interessante existente na linguagem, que é o recurso de apelido, ou alias. Este recurso serve para dar apelidos para as tabelas com nomes grandes, facilitando

referenciá-las dentro do comando.

Exemplo:

Comando de união utilizando o apelido P1 para a Tabela de Produto 1 e o apelido P2 para a Tabela de produto 2:

Tabela PRODUTO

| Nome Produto | Preço Unitário (R\$) |
|------------------|----------------------|
| Caneta Azul Pic | 1,2 |
| Régua 30cm | 0,9 |
| Caderno 50folhas | 1,5 |

Tabela PRODUTO2

| Nome Produto | Preço Unitário (R\$) |
|---------------------|----------------------|
| Caneta Vermelha Pic | 1,2 |
| Régua 30cm | 0,9 |
| Borracha branca | 0,8 |

```
SELECT P1.Nome, P1.Preco  
FROM PRODUTO1 P1  
UNION  
SELECT P2.Nome, P2.Preco  
FROM PRODUTO2 P2
```

Repare que os apelidos também podem ser utilizados para nomear campos além de tabelas.

Veja o exemplo a seguir, que dá um apelido para o campo de preço unitário do produto e lista os produtos que começam com a letra “C”:

```
SELECT PrecoUnitario AS Preco  
FROM PRODUTO  
WHERE Nome like 'C%'
```

[**Início da atividade online**](#)

Atividade 5 - Atende ao objetivo 3

Refaça a atividade 4 escrevendo os comandos SQL dos itens a,b e c, utilizando os seguintes apelidos para as tabelas:

- L – Para a tabela de lançamentos
- G – Para a tabela de gêneros de filme
- F – Para a tabela de filiais da locadora

Fim da atividade online

Conclusão

A linguagem SQL possui um vasto arsenal de comandos à nossa disposição, tanto para agrupar, sumarizar e totalizar informações quanto para integrar ou relacionar os dados das tabelas. É possível combinar um grande conjunto de instruções ou cláusulas, de forma a obter os dados relevantes nas consultas SQL. Nesta aula foram apresentadas as possibilidades mais comuns, mas existem muitas outras opções menos utilizadas, mas de igual importância para o aprendizado do aluno na linguagem.

Resumo

A linguagem SQL possui algumas palavras-chave chamadas de funções.

Funções de Agregação numéricas em SQL:

COUNT(*) : Exibe o número de linhas ou registros na tabela.

COUNT(nome_campo) : Exibe o número de linhas na coluna nome_campo. Se o valor for NULL ele não é contado.

SUM(nome_campo) : Soma os valores de uma coluna em todas as linhas ou registros.

AVG(nome_campo) : Média dos valores da coluna por todas as linhas.

MAX(nome_campo) : Valor máximo na coluna em todas as linhas.

MIN(nome_campo) : Valor mínimo na coluna em todas as linhas.

GROUP BY: Indica os nomes dos campos a partir dos quais os valores serão agrupados e totalizados ou sumarizados

HAVING: serve para acrescentar filtros ou condições em cima dos campos envolvidos nos agrupamentos determinados pela cláusula GROUP BY.

As operações de conjunto são:

- União - permite extrair todos os registros inclusos em Tabela1 **ou** Tabela2;
- Diferença - permite extrair todos os registros em Tabela1 que **não** estão em Tabela2;
- Interseção - permite extrair todos os registros que estão ao mesmo tempo em

Tabela1 e Tabela2;

- Produto Cartesiano - permite combinar todos os registros de Tabela1 com cada registro de Tabela2, produzindo como resultado Tabela1 x Tabela2 registros;

Parâmetro de Apelido AS (Alias) - Serve para dar apelidos para as tabelas com nomes grandes, facilitando referenciá-las dentro do comando.

Informações sobre a próxima aula

Uma visão é qualquer relação que não faz parte do modelo lógico do banco de dados, mas que é visível ao usuário, como uma relação virtual. Por outro lado, funções são muito importantes, pois permite que o desenvolvedor tenha maior flexibilidade facilitando a sua rotina no dia-a-dia.

Na próxima aula, você irá estudar sobre os conceitos de Visões e Funções num Banco de Dados.

Referências Bibliográficas

Date C.J., 2003. Introdução a Sistemas de Bancos de Dados. 8ed. Americana. Rio de Janeiro. Elsevier.

Carvalho C.R., 2006. SQL-Guia Prático. 2ed. Rio de Janeiro. Brasport.

Elmasri R., Navathe S., 2009. Sistemas de Banco de Dados. 5ed. São Paulo. Pearson Addison Wesley.

Heuser C.A. 2009. Projeto de Banco de Dados. 4ed. Porto Alegre. Bookman.

Setzer V.W. & Corrêa da Silva F.S. 2005. Bancos de Dados. 1ed. São Paulo. Edgard Blucher.

Silberschatz A., Korth H., 2008. Sistema de Banco de Dados. 3ed. São Paulo. Pearson Makron Books.

Aula 11

Views e Functions

Meta

Apresentar comandos importantes na manipulação de dados utilizando a linguagem padrão SQL.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Visões (Views)

- ✓ Definir o que é uma visão
- ✓ Criar uma visão
- ✓ Alterar uma visão
- ✓ Inserir a estrutura de uma visão **?????**
- ✓ Excluir uma visão

2. Funções (Functions)

- ✓ Definir o que é uma função
- ✓ Criar uma função
- ✓ Alterar uma função
- ✓ Exibir a estrutura de uma função **?????**
- ✓ Excluir uma função

Pré-requisitos

Para se ter um bom aproveitamento desta aula, é importante você relembrar os comandos DDL e DML aprendidos na aulas 8, 9 e 10 respetivamente. **???**

Introdução

Muitas vezes gostaríamos de utilizar os dados contidos em nosso banco de dados num formato diferente daquele em que realmente estão. Consideremos, como exemplo, uma situação onde constantemente queremos consultar o título do livro, seu preço, o nome da editora e a descrição do assunto do livro. Estas informações podem estar espalhadas em dois ou três tabelas. Se desejamos uni-las devemos realizar o comando SELECT com JOIN entre as tabelas. Entretanto, como a consulta é realizada constantemente, gostaria de ter uma diferente visão do nosso banco. A linguagem SQL nos permite isso, através de

um tipo de objeto chamado visão (ou views). O MySQL é um banco de dados que oferece muitos recursos aos desenvolvedores; entre tantos recursos temos as funções que são procedimentos já prontos para manipular os resultados da sua consulta vistos na aula anterior ou funções que podem ser criadas pelo próprio usuário. Apresentaremos funções que podem ser criadas pelos usuários. Vamos nessa!!!

Fim da Introdução

VISÃO ou VIEWS

Não é desejável que todos os usuários tenham acesso a todo o esquema conceitual, por isso que visões precisam ser definidas, mas o que é uma visão?

Visões são tabelas virtuais ou temporárias cujo conteúdo provém de tabelas reais. Ela não faz parte do esquema conceitual mas é visível a um grupo de usuários. Cada vez que realizamos uma consulta sobre a visão, o SGBD se encarrega de coletar dados nas tabelas de origem, a partir do comando SELECT que define a visão, e de atender à consulta que realizamos sobre a visão como se ela fosse uma tabela. O catálogo do SGBD é o repositório que armazena as definições das visões. Por outro lado, podemos ter muitas visões destinadas a diferentes usuários.

Vantagens na utilização de uma visão:

- ▲ Podemos controlar o acesso de alguns usuários a dados. Exemplo: divulgar o número de matrícula e a nota do aluno sem que seja necessário disponibilizar junto a essa tabela outros dados como CPF, endereço ou telefone.
- ▲ É possível combinar múltiplas tabelas e devolver aos usuários uma única tabela com os registros que ele deseja consultar.
- ▲ Visualizar dados gerais e omitir detalhes.
- ▲ Otimizar o tempo de acesso ao conjunto de dados. Após a utilização, o view fica numa tabela temporária e virtual e pode ser recuperado mais de uma vez.
- ▲ Simplificar o gerenciamento de permissões dos usuários, pois os mesmos poderão ter acesso a todas as tabelas virtuais, evitando assim, possíveis alterações feitas por qualquer um.

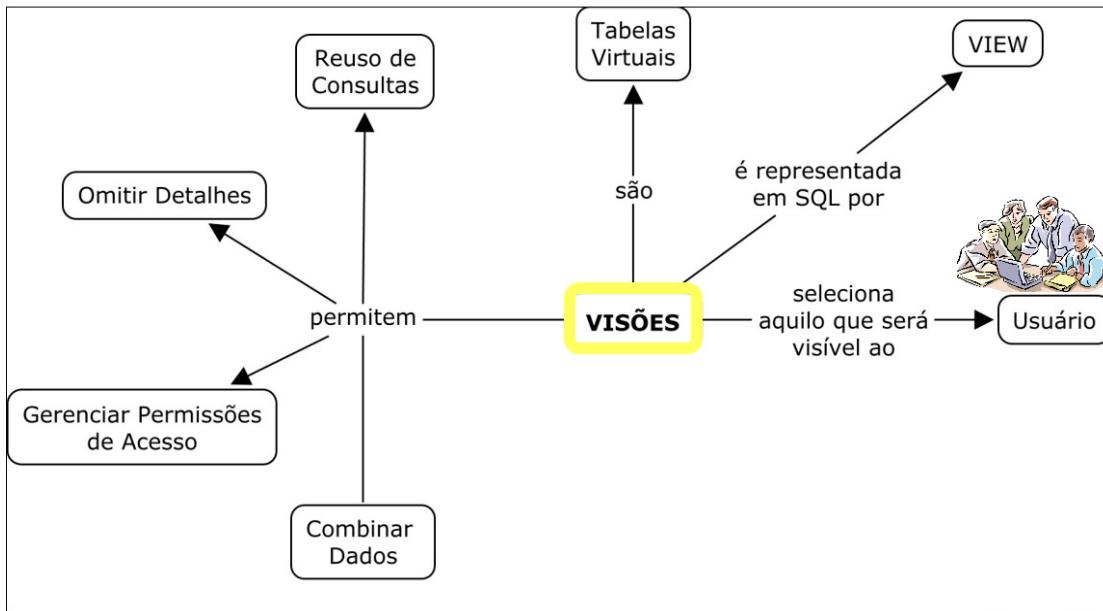


Figura 11.1 : Enfoque de Visão

Criando uma tabela de Visão

Uma visão possui nome, uma lista de atributos e uma consulta (query) que computa a visão. Com base na tabela criada com o comando CREATE TABLE você também pode criar uma tabela virtual que existe apenas quando é visualizada por um determinado

| Tabela EMP | | | | | | | | |
|------------|-------------|---------------|-----|------------|---------|------|----|--|
| Cod | Nome | Categoria | Num | Data | Salário | ADD | | |
| 01211 | João Silva | Presidente | 001 | 01/01/1990 | 15000 | 500 | 10 | |
| 11213 | Rita Santos | Administrador | 002 | 15/09/1995 | 9500 | 350 | 10 | |
| 11451 | Luiz Lua | Gerente | 003 | 07/12/1997 | 12000 | | 10 | |
| 11322 | Liz Bulnes | Gerente A | 005 | 13/10/1990 | 13500 | 420 | 10 | |
| 11310 | Ren Lula | Chefe 1 | 006 | 31/10/2000 | 10000 | 9200 | 20 | |
| 11412 | Tim Tah | Chefe 2 | 007 | 30/04/2005 | | | 20 | |
| 11412 | Ká Corrêa | Gerente B | 008 | 12/05/2010 | 8000 | | 20 | |

| View EMPV11 | | | | | | | | |
|-------------|------------|-----------|--|--|--|--|--|--|
| Cod | Nome | Categoria | | | | | | |
| 11451 | Luiz Lua | Gerente | | | | | | |
| 11322 | Liz Bulnes | Gerente A | | | | | | |
| 11211 | Ká Corrêa | Gerente B | | | | | | |

| | | | | | | | |
|-------|--------------|-----------|-----|------------|------|---|----|
| 12111 | Silvo Pontes | Chefe 3 | 020 | 12/04/1990 | 9000 | 0 | 30 |
| 12456 | Carla Caro | Técnico A | 040 | 01/01/2001 | 5500 | 0 | 30 |

usuário. Exemplo: Figura 11.2 criou-se a tabela virtual EMPV11 a partir da tabela EMP.

Figura 11.2 Tabela Virtual EMPV11

A sintaxe para criar uma visualização:

```
CREATE VIEW nome_tabela_virtual  
coluna1, coluna2, ... AS  
SELECT nome_atributo novoNome_Atributo  
FROM nome_tabela1, nome_tabela2, ...  
WHERE condição
```

onde:

- CREATE VIEW: comando que cria a tabela virtual ou define uma nova visão.
- *nova_tabela_virtual*: nome em que a visão poderá ser referenciada.
- coluna1, coluna2,...: indica quantas e quais as colunas que farão parte da view.
- SELECT: seleciona as colunas para serem visualizadas. O caractere * é utilizado para indicar que serão listadas todas as colunas das tabelas.
- novoNome_Atributo: campo opcional usado quando há colunas de mesmo nome em uma view.
- FROM *nome_tabela1, nome_tabela2, ...*: comando que utiliza os nomes das tabelas-base.
- WHERE: comando que expressa a condição da consulta.

Exemplo:

Considere as seguintes tabelas:

Tabela Empregado

| Código | Nome | Endereço | Setor | Salário |
|--------|-------|----------------------|-------|---------|
| 01 | Lucas | Rua F 12 DQ-RJ | A | 1200,90 |
| 02 | Maria | Rua Alagoas 34 TJ-RJ | B | 850,15 |
| 03 | João | Rua Pedra 11 LB-RJ | C | 560,22 |
| 04 | Pedro | Rua Ave 763 CP-RJ | B | 925,40 |

Figura 11.3 Tabela Empregado

Tabela Produto

| Código | Nome | Preço | Setor |
|----------|-----------|--------|-------|
| 12341111 | Produto 1 | 123,99 | A |
| 34511200 | Produto 2 | 78,99 | C |
| 67349121 | Produto 3 | 235,99 | B |

Figura 11.4 Tabela Produto

- a) Crie uma visão que possua todos os empregados do setor B e nomes dos produtos que são capazes de produzir.

```

CREATE VIEW ProducaoB AS
SELECT E.* , P.Nome AS NomeProduto
FROM Empregado E, Produto P
WHERE E.Setor = P.Setor AND
      E.Setor = 'B';

```

Visualizando o conteúdo da tabela virtual criada: SELECT * FROM ProducaoB;

| Código | Nome | Endereço | Setor | Salário | NomeProduto |
|--------|-------|-----------------|-------|---------|-------------|
| 02 | Maria | Rua Alagoas ... | B | 850,15 | Produto 3 |
| 04 | Pedro | Rua Ave ... | B | 925,40 | Produto 3 |

b) Crie a visualização chamada Tb1V considerando Código, nome e salário do empregado, cujos salários sejam maiores que R\$900:

```

CREATE VIEW Tb1V AS
SELECT Cod_Emp, Nome_EMP, Salario
FROM EMPREGADO
WHERE Salario > 900;

```

Visualizando o conteúdo da tabela virtual criada: SELECT * FROM Tb1V;

| Código | Nome | Salário |
|--------|-------|---------|
| 01 | Lucas | 1200,90 |
| 04 | Pedro | 925,40 |

c) Crie a visualização a partir da tabela virtual Tb1V (criada no item anterior) chamada Tb2V considerando os atributos Nome e Salário.

```

CREATE VIEW Tb2V AS
SELECT Nome, Salario
FROM Tb1V ;

```

Visualizando o conteúdo da tabela virtual criada: SELECT * FROM Tb1V;

| Nome | Salário |
|-------|---------|
| Lucas | 1200,90 |
| Pedro | 925,40 |

d) Crie a visualização baseada na tabela PRODUTO considerando os atributos Código,

Nome, Preço e que sejam do terceiro setor.

```
CREATE VIEW V3Setor AS  
SELECT Cod_Produto, Nome_Produto, Preco_Produto  
FROM PRODUTO  
WHERE setor = 3;
```

Visualizando o conteúdo da tabela virtual criada: `SELECT * FROM V3Setor;`

| Código | Nome | Preço |
|----------|-----------|--------|
| 12341111 | Produto 1 | 123,99 |
| 67349121 | Produto 3 | 235,99 |

Views podem ser criadas facilmente, como vimos anteriormente. Mas para termos de atualização tais como UPDATE, INSERT e DELETE, que de fato alteram as tabelas base, temos que ter alguns cuidados na criação do objeto de visualização. Vejamos como é feito isso no seguinte tema.

Atualizando uma View

Uma View criada com funções agregadas, por exemplo, não poderá receber atualizações, pois os dados logicamente estão agregados ou agrupados e não teremos correspondências diretas para uma exclusão ou atualização. Então Views são úteis em consultas, mas existem restrições em relação a atualizações.

Quando temos uma View com um SELECT simples, sem utilizar agrupamentos, podemos atualizar as tabelas base que tem seus dados mapeados para esta view.

UPDATE

Criarmos a tabela DVDFilmes na qual utilizaremos como tabela base:

| Código | Autor Principal | Autor Secundário | Tipo | Ano | Copias |
|--------|-----------------|------------------|---------|------|--------|
| 0121 | Robert Smith | Mary Kliman | Ficção | 2009 | 20 |
| 0454 | Rose Wear | Fredy Wuilliams | Romance | 2011 | 20 |
| 5654 | Jean Pasteur | Peter Miyagusku | Ação | 2010 | 10 |
| 3227 | Klean Kus | Any Santee | Romance | 2008 | 15 |

Figura 11.5 Tabela DVDFilmes

Com a tabela base DVDFilmes criada, vamos definir a View chamada Vdvd01 para que possamos atualizar a mesma.

```
CREATE VIEWS Vdvd01 AS  
SELECT codigo, ano, copias
```

```
FROM DVDFilmes  
WHERE tipo = 'Romance';
```

Visualizando a tabela virtual Vdvd01: SELECT * FROM Vdvd01;

| Código | Ano | Copias |
|--------|------|--------|
| 0454 | 2011 | 20 |
| 3227 | 2008 | 15 |

Agora, vamos atualizar o número de cópias do filme de Rose Wear, ou seja, atualização no código de filme 0454:

```
UPDATE Vdvd01 SET copias = copias +1  
WHERE codigo = 0454;
```

Visualizando a tabela virtual atualizada Vdvd01: SELECT * FROM Vdvd01;

| Código | Ano | Copias |
|--------|------|--------|
| 0454 | 2011 | 21 |
| 3227 | 2008 | 15 |

INSERT

Devemos utilizar os comandos DML, como em qualquer outra tabela do banco de dados.
Exemplo: Considere a tabela Vdvd01:

```
SELECT * FROM Vdvd01;
```

| Código | Ano | Copias |
|--------|------|--------|
| 0454 | 2011 | 20 |
| 3227 | 2008 | 15 |

Desejamos incluir o código do filme 1231 na tabela Vdvd01:

```
INSERT INTO Vdvd01 (  
codigo, ano, copias)  
VALUES (  
1231, 2011, 12);
```

Visualizando a tabela virtual atualizada Vdvd01 atualizada: SELECT * FROM Vdvd01;

| Código | Ano | Copias |
|--------|------|--------|
| 0454 | 2011 | 20 |
| 3227 | 2008 | 15 |
| 1231 | 2011 | 12 |

DELETE

Para apagarmos uma View, basta utilizar o comando:

```
DROP VIEW nome_tabela_virtual;
```

Sabemos que uma tabela virtual permanece no banco de dados, de forma que a visão pode ser acessada em qualquer momento, mas agora imagine que precisamos apagar a tabela virtual Vdvd01 criada anteriormente; procederemos da seguinte forma:

```
DROP VIEW Vdvd01;
```

Na exclusão da visão, somente sua definição é excluída, pois os dados permanecem nas tabelas originais.

Início da atividade

Atividade 1- Atende ao objetivo 1

(1) Considere as seguintes tabelas:

Tabela Candidato

| CPF | RG | Nomes | Sexo | Endereço | Telefone |
|-------------|---------|-----------------|------|----------------------|----------|
| 05528387728 | 3326888 | Rita dos Santos | F | Rua Bahia 11 PT-RJ | 22351359 |
| 02343456621 | 2126987 | Lúcia Pereira | F | Rua Santos 34 DQ-RJ | 24134444 |
| 12345678911 | 1526912 | Tiago Silva | M | Rua Rosa 1 IT-SP | 23121212 |
| 72516051219 | 6426411 | Marco Oliveira | M | Rua G 781 AL-SC | 25891911 |
| 03256712355 | 4846978 | Ana Moura | F | Rua Brasil 78 -CG-RJ | 31453900 |

Tabela Aprovado

| CPF | Nomes | Nota |
|-------------|-----------------|-------|
| 02343456621 | Lúcia Pereira | 100,0 |
| 12345678911 | Tiago Silva | 98,7 |
| 03256712355 | Ana Moura | 95,1 |
| 05528387728 | Rita dos Santos | 90,0 |

- a) Crie uma visão consistindo de todos dados dos candidatos aprovados cuja nota seja maior ou igual a 95,0, exibindo também o campo da nota além dos dados cadastrais.
- b) Crie uma nova tabela com o nome VisaoAB, selecionando os nomes dos aprovados da tabela Aprovado.
- c) Crie uma visualização chamada VisaoXF dos candidatos de sexo feminino, considerando os campos a serem visualizados nome, endereço e telefone.
- d) Utilize a tabela base VisaoXF e atualize o endereço de Ana Moura sendo o endereço atual Rua Jardins 123 TJ-SP. Mostre a tabela virtual atualizada.

- e) Inclua o aluno aprovado Alexandre Silva dos Santos na tabela VisaoAB.
f) Exclua as tabelas virtuais criadas no item b), c) e d).

Diagramação, inserir 10 linhas para a resposta.

Resposta Comentada

a) CREATE VIEW VisaoAP as

SELECT *

FROM Aprovados

WHERE nota > = 95,0;

Visualizando o conteúdo da tabela virtual : SELECT * FROM VisaoAP;

| CPF | Nomes | Nota |
|-------------|---------------|-------|
| 02343456621 | Lúcia Pereira | 100,0 |
| 12345678911 | Tiago Silva | 98,7 |

b) CREATE VIEW VisaoAB as

SELECT nome

FROM Aprovados;

Assim, podemos visualizar o conteúdo da tabela virtual criada escrevendo o comando abaixo:

SELECT * FROM VisaoAB;

| Nomes |
|-----------------|
| Lúcia Pereira |
| Tiago Silva |
| Ana Moura |
| Rita dos Santos |

c) CREATE VIEW VisaoXF as

SELECT nome, endereço, telefone

FROM Candidato;

WHERE sexo = 'F';

Visualizando o conteúdo da tabela virtual VisãoXF : SELECT * FROM VisaoXF;

| Nomes | Endereço | Telefone |
|-----------------|--------------------|----------|
| Rita dos Santos | Rua Bahia 11 PT-RJ | 22351359 |

| | | |
|---------------|----------------------|----------|
| Lúcia Pereira | Rua Santos 34 DQ-RJ | 24134444 |
| Ana Moura | Rua Brasil 78 -CG-RJ | 31453900 |

As tabelas virtuais criadas VisaoAP, VisaoAB e VisaoXF foram acopladas ao banco de dados.

d) Atualizando o endereço de Ana Moura:

```
UPDATE VisaoXF SET endereco = 'Rua Jardins 123 TJ-RJ '
WHERE nome = 'Ana Moura';
```

Visualizando a tabela virtual atualizada VisaoXF: SELECT * FROM VisaoXF;

| Nomes | Endereço | Telefone |
|-----------------|-----------------------|----------|
| Rita dos Santos | Rua Bahia 11 PT-RJ | 22351359 |
| Lúcia Pereira | Rua Santos 34 DQ-RJ | 24134444 |
| Ana Moura | Rua Jardins 123 TJ-RJ | 31453900 |

e) Incluindo os dados do aluno na tabela VisaoAB:

```
INSERT INTO VisaoAB (
nomes) VALUES ( Alexandre Silva dos Santos);
```

Visualizando a tabela virtual atualizada: SELECT * FROM VisaoAB;

| Nomes |
|----------------------------|
| Lúcia Pereira |
| Tiago Silva |
| Ana Moura |
| Rita dos Santos |
| Alexandre Silva dos Santos |

f) Excluindo as tabelas virtuais VisaoAP, VisaoAB e VisaoXF:

```
DROP VIEW VisaoAP;
DROP VIEW VisaoAB;
DROP VIEW VisaoXF;
```

Fim da atividade

Funções

Podemos dividir as funções escritas pelos usuários em três categorias:

- ▲ Funções de atualizações de banco de dados: Operações de mudança SQL (insere, atualiza, seleciona e exclui).

- ▲ Funções de computação: Podem incorporar a lógica if-then-else e várias operações.
- ▲ Funções de pesquisa: Executam consultas com o banco de dados.

As funções SQL executam uma lista arbitrária de declarações SQL, retornando o resultado da última consulta da lista. Caso a última consulta não retorne nenhuma linha, é retornado o valor nulo.

CRIAR

Para criar uma função devemos seguir a seguinte sintaxe:

```
CREATE FUNCTION nome_função(parâmetros tipos )
RETURNS tipo
corpo da função;
```

onde:

CREATE FUNCTION : instrução usada em versões novas do MySQL.

nome_função : nome da função a ser criada.

parâmetros: A lista de parâmetros entre parenteses deve estar sempre presente. Se não houver parâmetros, uma lista de parâmetros vazia de () deve ser usada.

RETURNS tipo: pode ser especificada apenas por uma FUNCTION. É usada para indicar o tipo de retorno da função, e o corpo da função deve conter uma instrução RETURN valor.

Tipo: STRING ou REAL ou INTEGER.

corpo da função: O corpo de uma função SQL deve ser uma lista contendo uma ou mais declarações SQL separadas por ponto e vírgula (;). O ponto e vírgula após a última declaração é opcional.

Observações:

- (1) Qualquer coleção de comandos na linguagem SQL pode ser juntada e definida como uma função. Além de comandos SELECT, podem existir comandos de manipulação de dados (INSERT, UPDATE e DELETE), assim como outros comandos SQL.
- (2) Entretanto, o comando final deve ser um SELECT retornando o que foi especificado como sendo o tipo retornado pela função. Como alternativa, se for desejado definir uma função SQL que realiza ações mas não retorna um valor útil, a função pode ser definida como retornando void.

Naturalmente de nada adiantaria existirem esses comandos se não houvesse comandos

de controle de fluxo e repetição. Resumidamente, vamos comentar cada um deles.

IF ... THEN ... ELSE ... END IF

Estrutura condicional se/não(if/else) responsável pelo desvio condicional com base no teste de condição da cláusula IF. Esta estrutura impõe condições na execução de uma instrução. Sintaxe:

```
SELECT if ( condição2, ação 1, ação 2) AS nome_coluna FROM nome_tabela.
```

onde:

O if acima verifica a condição, se for verdadeira executa a ação1, senão executa ação2.

Ou podemos usar if/else da seguinte maneira:

```
if condição then
    Ação 1;
    Ação N;
else
    Ação 12;
    Ação N2;
end if;
```

onde:

Caso a *condição* seja verdadeira, as ações após a cláusula **then** serão executadas. Caso não seja verdadeira, as ações após o **else** serão executadas. Dependendo da implementação podemos colocar outros **If's** dentro do comando **then** ou **else**, neste caso teremos if's aninhadas.

WHILE

Este comando é usado para a execução repetida de um bloco de instruções. As instruções serão executadas repetidamente com base da condição especificada seja verdadeira. A instrução **while** pode ser controlada internamente com os comandos **break** e **continue**. O comando **break** provoca uma saída do loop **while** mais interno e o comando **continue** faz com que o loop **while** seja reiniciado, ignorando as demais instruções depois do **continue**.

Sintaxe:

```
while (condição) do
    { Ações | break | continue }
```

Exemplo:

1. A seguir temos um exemplo de uma função chamada hello que utiliza um parâmetro tipo caractere, realiza uma operação usando uma função SQL e retorna o resultado:

```
CREATE FUNCTION hello (s CHAR(20)) RETURNS CHAR(50)
RETURN CONCAT('Hello, ',s,'!');
```

CONCAT ==> trata-se de uma função do MySql e é utilizado quando queremos unir o resultado de dois campos em um só ou unir campos junto com uma cadeia de caracteres.

2. Digamos que temos uma coluna em nossa base de dados que seja do tipo inteiro, que armazena o valor 0 e 10. Você gostaria utilizando uma função que receba como parâmetro aquele número e que o resultado fosse uma string(cadeia de caracteres) mais informativa, como por exemplo 'aprovado' e 'reprovado'.

```
CREATE FUNCTION Prova (nota INT)
RETURNS VARCHAR(10)

BEGIN
DECLARE resultado VARCHAR(50);

IF nota > 6 THEN SET resultado = 'Aprovado';
ELSE
    SET resultado = 'Reprovado';
END IF;

RETURN resultado;
END;
```

A instrução BEGIN ... END, é utilizada para agrupar um conjunto de instruções .

3. Crie uma função chamada Comparar para realizar a comparação de dois números inteiros e o resultado ou retorno seja a seguinte mensagem “numero 1 foi igual/maior/menor numero2” :

```
CREATE FUNCTION Comparar (num1 INT, num2 INT)
RETURNS VARCHAR(50)

BEGIN
DECLARE resultado VARCHAR(50);

IF num1 = num2 THEN SET resultado = 'igual';
ELSE
    IF num1 > num2 THEN SET resultado = 'maior';
    ELSE SET resultado = 'menor';
END IF;

SET resultado = CONCAT('foi ', resultado, ' que');
END IF;
```

```
SET resultado = CONCAT(num1, ' ', resultado, ' ', num2, '.');
RETURN resultado;
END;
```

4. Crie uma função chamada somar que possua dois parâmetros tipos inteiros e que o retorno seja também inteiro.

```
CREATE FUNCION somar(n1 integer, n2 integer) RETURNS integer
BEGIN
    SELECT n1 + n2 INTO n1;
    RETURN n1;
END;
```

5. Um funcionário observou em seu contracheque que o salário era negativo; então crie uma função chamada limpar_salario que remova as linhas contendo salários negativos da tabela EMP:

```
CREATE FUNCION limpar_salario(codigo_emp INT) RETURNS INT
DELETE FROM EMP
WHERE EMP.salario < 0 AND EMP.codigo = codigo_emp;
```

Mostrando os resultados: `SELECT limpar_salario(codigo_emp);`

| |
|----------------------------|
| limpar_salario(codigo_emp) |
| 1 |

6. Criar uma função que calcula a média de quatro notas de um aluno da tabela Notas:

```
CREATE FUNCTION media (nome VARCHAR(10)) RETURNS FLOAT
BEGIN
    DECLARE n1,n2,n3,n4 INT;
    DECLARE med FLOAT;
    SELECT nota1,nota2,nota3,nota4 INTO n1,n2,n3,n4 FROM Notas WHERE aluno = nome;
    SET med = (n1+n2+n3+n4)/4;
    RETURN med;
END;
```

7. Crie uma função chamada debitar para realizar débitos numa conta corrente do banco,

atualize o saldo e mostre o saldo atual:

```
CREATE FUNCION debitard(conta integer, novo_saldo numeric) RETURNS numeric
    UPDATE Corrente
        SET Corrente.saldo = Corrente.saldo - novo_saldo
        WHERE Corrente.conta = conta;
    SELECT Corrente.saldo FROM Corrente WHERE Corrente.conta = conta;
```

8. Se desejamos saber as quantidades de estudantes de tipo masculino na tabela Aluno.

```
DECLARE contaSexM INT;
WHILE (codigo != 1 ) FROM Aluno
BEGIN
    IF (Aluno.sexo = 'M') THEN contaSexM = contaSexM +1;
END IF;
END
```

ALTERAR

ALTER FUNCTION nome_função [características ...]

onde:

- ▲ características: NAME nome_função
 SQL SECURITY {DEFINER | INVOKER}
 COMMENT string

Este comando pode ser usado para renomear uma função ou para alterar suas características. Mais de uma mudança pode ser especificada em uma instrução ALTER FUNCTION.

Exemplo:

1. Alterar a função criada anteriormente que calcula a média de quatro notas de um aluno: pela média que calcula duas notas

```
ALTER FUNCTION media
MODIFIED FUNCTION BODY
```

EXIBIR

O comando SHOW é uma extensão do MySQL. Ele retorna características da rotina, tais como nome, tipo, quem criou, datas de modificações e criação. Se nenhum padrão é especificado, a informação de todas as funções armazenadas é listado, dependendo de qual instrução você utiliza. Sintaxe:

SHOW FUNCTION STATUS [LIKE estrutura]

Exemplo:

```
SHOW FUNCTION STATUS LIKE 'tabela%'

Db: test
Name: tabela
Type: PROCEDURE
Definer: testuser123@localhost
Modified: 2011-08-07 13:09:11
Created: 2011-08-10 08:11:23
Security_type: DEFINER
Comment:
```

EXCLUIR

O comando para deletar uma função, ou seja, a rotina especificada seja removida do servidor é:

```
DROP FUNCTION [IF EXISTS] nome_função;
```

onde:

- ▲ A cláusula IF EXISTS é uma extensão do MySQL. Ela previne que um erro ocorra se a função não existe.
- ▲ nome_função: O nome da função existente.

Início da atividade

Atividade 2- Atende ao objetivo 2

Crie uma função que:

- 1) retorne o quadrado de um número (n^2), passado por parâmetro.
- 2) retorne o fatorial de um número, informando via parâmetro.
- 3) altere a nota de um aluno da tabela Histórico, passando por parâmetros código do aluno e o novo valor da nota.
- 4) Excluir as funções criadas nos itens 3).

Diagramação, inserir 10 linhas para a resposta.

Resposta Comentada

1) CREATE FUNCTION quadrado (numero integer) RETURNS integer
BEGIN

```
    SELECT numero * numero into numero;
    RETURN numero;
END;
```

2) CREATE FUNCTION fatorial (numero integer) RETURNS integer
BEGIN

```

if numero < = 1 then RETURN numero;
else
    SELECT factorial(numero-1)* numero into numero;
end if;
RETURN numero;
END;

```

- 3) CREATE FUNCION altera_nota(codigo integer, nota numeric) RETURNS numeric
 UPDATE Historico
 SET Historico.nota = nota
 WHERE Historico.cod = codigo;
 SELECT Historico.nota FROM Historico WHERE Historico.cod = codigo;
- 4) DROP FUNCTION IF EXISTS altera_nota;

Início da atividade online

Atividade 1 - Atende ao objetivo 2

Acesse o ambiente virtual e resolva a atividade a seguir em MySQL, enviando sua resposta ao tutor. Crie as seguintes funções com seus parâmetros e retornos respectivos. Seja uma tabela CtaCorrente que possui os atributos: id_cliente, nome_cliente, cpf_cliente, saldo e dtCadastro.

- a) Apagar as informações do cliente João Carlos Santos com o Cpf: 05528387728, identificação 011121212 e com data de cadastro 20/05/2011.
- b) Inserir as informações de um novo cliente se o nome e o cpf são diferentes de caracteres vazios.
- c) Atualizar o número do CPF de um cliente de um banco.
- d) Se os saldos dos clientes for negativo e a data do cadastro maior e igual 10/01/2011 sobre os saldos.
- e) Excluir as funções criadas nos itens anteriores.

Fim da Atividade Online

Conclusão

Vimos então como trabalhar com visualizações ou View que nada mais são que mapeamentos lógicos de outras tabelas em uma nova, definidas com comandos SELECT VIEW. Vimos como criar Views, como alterar ou sobrescrever as mesmas.

O MySQL é um banco de dados que oferece muitos recursos aos desenvolvedores, entre tantos recursos temos as funções. Que são procedimentos já prontos para manipular os resultados da sua consulta ou funções que podem ser criadas pelo próprio usuário. Na prática as funções são menos poderosas que os procedimentos, eles necessitam de um valor de retorno.

Resumo

A sintaxe para criar uma visualização:

```
CREATE VIEW nome_tabela_virtual  
    coluna1, coluna2, ... AS  
        SELECT nome_atributo novoNome_Atributo  
        FROM nome_tabela  
        WHERE condição
```

onde:

- **CREATE VIEW**: comando que cria a tabela virtual ou define uma nova visão.
- *nome_tabela_virtual*: nome que a visão poderá ser referenciada.
- coluna1, coluna2,...: indica quantas e quais as colunas que farão parte da view.
- **SELECT**: seleciona as colunas para serem visualizadas. O caractere * é utilizado para indicar que serão listadas todas as colunas das tabelas.
- novoNome_Atributo: campo opcional usado quando há colunas de mesmo nome em uma view.
- **FROM nome_tabela**: comando que utiliza o nome da tabela base.
- **WHERE**: comando que executa a condição de consulta;

Para apagarmos uma View, basta utilizar o comando:

```
DROP VIEW nome_tabela_virtual;
```

Funções

Podemos dividir as funções escritas pelos usuários em três categorias:

- ▲ Funções de atualizações de banco de dados: Operações de mudança SQL (insere, atualiza, seleciona e exclui).
- ▲ Funções de computação: Podem incorporar a lógica if-then-else e várias operações.

- ▲ Funções de pesquisa: Executam consultas com o banco de dados.

As funções SQL executam uma lista arbitrária de declarações SQL, retornando o resultado da última consulta da lista. Caso a última consulta não retorne nenhuma linha, é retornado o valor nulo.

CRIAR

Para criar uma função devemos seguir a seguinte sintaxe:

```
CREATE FUNCTION nome_função(parâmetros tipos )
RETURNS tipo
corpo da função;
```

onde:

CREATE FUNCTION : instrução usada em versões novas do MySQL.

nome_função : nome da função a ser criada.

parâmetros: A lista de parâmetros entre parenteses deve estar sempre presente. Se não houver parâmetros, uma lista de parâmetros vazia de () deve ser usada.

RETURNS tipo: pode ser especificada apenas por uma FUNCTION. É usada para indicar o tipo de retorno da função, e o corpo da função deve conter uma instrução RETURN valor.

Tipo: STRING ou REAL ou INTEGER.

corpo da função: O corpo de uma função SQL deve ser uma lista contendo uma ou mais declarações SQL separadas por ponto e vírgula (;). O ponto e vírgula após a última declaração é opcional.

Observações:

- (1) Qualquer coleção de comandos na linguagem SQL pode ser juntada e definida como uma função. Além de comandos SELECT, podem existir comandos de manipulação de dados (INSERT, UPDATE e DELETE), assim como outros comandos SQL
- (2) Entretanto, o comando final deve ser um SELECT retornando o que foi especificado como sendo o tipo retornado pela função. Como alternativa, caso haja necessidade de se definir uma função SQL que realiza ações mas não retorna um valor útil, a função poderá ser definida como retornando void.

EXIBIR

O comando SHOW é uma extensão do MySQL. Ele retorna características da rotina, tais como nome, tipo, quem criou, datas de modificações e criação. Se nenhum padrão é

especificado, a informação de todas as funções armazenadas é listado, dependendo de qual instrução você utiliza. Sintaxe:

SHOW FUNCTION STATUS [LIKE estrutura]

EXCLUIR

O comando para deletar uma função, ou seja, a rotina especificada seja removida do servidor é:

DROP FUNCTION [IF EXISTS] nome_função;

onde:

- ▲ A cláusula IF EXISTS é uma extensão do MySQL. Ela previne que um erro ocorra se a função não existe.
- ▲ nome_função: O nome da função existente.

Informações sobre a próxima aula

Normalização é o processo formal passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros. Uma regra de ouro que devemos observar quando do projeto de um Banco de Dados baseado no Modelo Relacional de Dados é a de "não misturar assuntos em uma mesma Tabela". Por exemplo: na Tabela Clientes devemos colocar somente campos relacionados com o assunto Clientes. Não devemos misturar campos relacionados com outros assuntos, tais como Pedidos, Produtos, etc. Essa "Mistura de Assuntos" em uma mesma tabela acaba por gerar repetição desnecessária dos dados bem como inconsistência dos dados.

Normalmente após a aplicação das regras de **normalização de dados**, algumas tabelas acabam sendo divididas em duas ou mais tabelas, o que no final gera um número maior de tabelas do que o originalmente existente. Este processo causa a simplificação dos atributos de uma tabela, colaborando significativamente para a estabilidade do modelo de dados, reduzindo-se consideravelmente as necessidades de manutenção.

Na próxima aula, você irá estudar sobre esses conceitos importantes num Banco de Dados.

Referências Bibliográficas

Date C.J., 2003. Introdução a Sistemas de Bancos de Dados. 8ed. Americana. Rio de Janeiro. Elsevier.

Carvalho C.R., 2006. SQL-Guia Prático. 2ed. Rio de Janeiro. Brasport.

Elmasri R., Navathe S., 2009. Sistemas de Banco de Dados. 5ed. São Paulo. Pearson

Addison Wesley.

Heuser C.A. 2009. Projeto de Banco de Dados. 4ed. Porto Alegre. Bookman.

Setzer V.W. & Corrêa da Silva F.S. 2005. Bancos de Dados. 1ed. São Paulo. Edgard Blucher.

Silberschatz A., Korth H., 2008. Sistema de Banco de Dados. 3ed. São Paulo. Pearson Makron Books.

Schwartz Baron. 2009. Alto Desempenho em MySQL. 2ed. Rio de Janeiro. Alta Books

Curso de Extensão: Modelando e Implementando Bancos de Dados Relacionais
Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 12

Processo de Normalização

Meta

Fornecer um apoio para o projeto de banco de dados relacional para a tarefa de normalização de um modelo de banco de dados.

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Compreender o conceito e importância sobre a Normalização.
2. Aplicar a Primeira Forma Normal (1FN).
3. Aplicar a Segunda Forma Normal (2FN).
4. Aplicar a Terceira Forma Normal (3FN).
5. Aplicar a Terceira Forma Normal Estendida (BCNF).

Pré-requisitos

Para se ter um bom aproveitamento desta aula, é importante lembrar os conceitos de atributos, atributos multivalorado, chaves candidatas, compostas, primárias e entidades, referentes à aula 2 até aula 5.

Introdução

Porque devemos normalizar?

É necessário normalizar dados para gerenciar apropriadamente um banco de dados relacional. Consequentemente, é preciso dividir a tabela.

O processo de normalização consiste em regras de ouro em cada uma das etapas; o conceito principal é de "não misturar assuntos em uma mesma Tabela". Por exemplo: na Tabela Clientes devemos colocar somente campos relacionados com o assunto Clientes. Não devemos misturar campos relacionados com outros assuntos, tais como

Pedidos, Produtos, etc. Essa "Mistura de Assuntos" em uma mesma tabela acaba por gerar repetição desnecessária dos dados bem como inconsistência dos dados.

Normalmente após a aplicação das regras de **normalização de dados**, algumas tabelas acabam sendo divididas em duas ou mais tabelas, o que no final gera um número maior de tabelas do que o originalmente existente. Este processo causa a simplificação dos atributos de uma tabela, colaborando significativamente para a estabilidade do modelo de dados, reduzindo-se consideravelmente as necessidades de manutenção. Nesta aula, você irá estudar sobre esses conceitos importantes.

Fim da Introdução

O que será normalização?

Normalização é o processo formal passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros.

Normalização de Dados consiste em definir o formato lógico adequado para as estruturas de dados identificados no projeto lógico do sistema, com o objetivo de minimizar o espaço utilizado pelos dados e garantir a integridade e confiabilidade das informações.

Normalização - Benefícios

Dentre vários pontos como benefícios do processo de normalização vamos descrever alguns.

1. Estabilidade do Modelo Lógico

- Entende-se por estabilidade a capacidade de um modelo manter-se inalterado diante a mudanças que venham a ser percebidas ou introduzidas no mini-mundo modelado.
- A implementação de tabelas não normalizadas compromete o próprio modelo lógico, ou seja, problemas nas estruturas de armazenamentos, alocação de espaços, etc. Afetando em menor ou maior grau o desempenho e a produtividade do desenvolvimento de aplicações.

2. Flexibilidade

- Por flexibilidade entendemos a capacidade de adaptação a demandas diferenciadas, a expansão e redução ou omissão ou presença. Se numa

estrutura de dados implementada, não tivermos a devida flexibilidade, estaremos impondo restrições de uso e tornando as estruturas de dados e de processos dependentes de limites rígidos.

- Através da flexibilidade => Aumenta o grau de adaptabilidade do banco de dados em função de mudanças no Mundo Real (Regras de Negócio e Requisitos de Informação)

3. Integridade

- As estruturas de dados obtidas pelo processo de modelagem necessitam de recursos para que os dados a serem armazenados possam ter qualidade. A qualidade de um dado está vinculada: atualidade, veracidade, fidelidade, integridade, etc. A integridade diz respeito à qualidade do dado.
- Se um dado aparece mapeado em mais de um local de modo diferente, poderemos ter indícios de que não há integridade entre eles.
- Através do processo de normalização a não integridade (anomalia) é eliminada no modelo.

4. Economia

- Os custos de armazenamento, gerados pela redundância, pela desnormalização e por outras anomalias presentes na construção das tabelas são os maiores custos existentes.
- Esse custo representa todo e qualquer esforço, tempo, ou valor agregado ao fato de manipulação volumes de dados maiores do que os efetivamente necessários.
- Tempos envolvidos em processamento, discos, mídias para backup, local para armazenamento e outros podem ter seus custos aumentados.
- Manter processos redundantes teremos custos adicionais como o caso de atualização dos dados.
- Através do processo de normalização, minimiza os custos de qualquer esforço, tempo e armazenamento.

5. Expressividade

- Neste caso trata-se de um aspecto subjetivo, pois procura-se obter a máxima representatividade entre o ambiente observado e o modelo de banco de dados.
- Através do processo de normalização, aumenta o grau de representatividade do

banco de dados tornando os "objetos do banco de dados" mais próximos dos objetos, respectivos, do Mundo Real.

- E também aumenta o grau de consistência das informações armazenadas no banco de dados.

Início da atividade

Atividade 1 - Atende ao objetivo 1

Descreva em forma resumida os benefícios do processo de normalização.

Diagramação, inserir um espaço de 10 linhas para cada item de resposta.

Resposta Comentada

- Estabilidade do Modelo Lógico: capacidade de um modelo manter-se inalterado face a mudanças que venham acontecer.
- Flexibilidade: aumenta o grau de adaptabilidade do banco de dados em função de mudanças no Mundo Real (Regras de Negócio e Requisitos de Informação)
- Integridade: elimina as redundâncias dos dados na tabela.
- Economia: minimiza os custos de qualquer esforço, tempo e armazenamento.
- Expressividade: aumenta o grau de consistência das informações armazenadas no banco de dados.

Fim da atividade

Formas Normais

A normalização é feita através da análise dos dados que compõem as estruturas utilizando o conceito chamado “Formas Normais (FN)”. As FN são conjuntos de restrições nas quais os dados devem satisfazê-las.

Exemplo: pode-se dizer que a estrutura está na primeira forma normal (1FN), se os dados que a compõem satisfazem as restrições definidas para esta etapa.

A normalização completa dos dados é feita seguindo as restrições das três formas normais existentes, sendo que a passagem de uma FN para outra é feita tendo como base o resultado obtido na etapa anterior, ou seja, na FN anterior.

Para realizar a normalização dos dados, é primordial que seja definido um campo chave para a estrutura, campo este que permite identificar os demais campos da estrutura. Toda entidade do banco de dados possui um atributo ou um conjunto de atributos do tipo chave primária já identificados durante a etapa de modelagem das

entidades e relacionamentos, nesta forma veremos as diferentes formas de normalização.

As formas normais são:

Primeira Forma Normal (1FN)

Regra 1FN: Cada atributo de uma tabela deve ter apenas um único valor.

Devemos,

1. transformar os atributos em atributos atômicos;
2. extrair com os grupos de dados repetidos.

Exemplo:

Como estudo, iremos considerar a tabela de Acessórios vendidos:

| Código | Tipo | Forma | Marca | Material |
|--------|---------|------------|----------|---------------------------|
| 001 | brinco | losango | M&M | prata, ouro 18K |
| 003 | anel | redonda | Stylus1 | ouro 18k, prata, diamante |
| 005 | efígie | retangular | R12Z | Ouro18k, rubi, perola |
| 008 | relógio | quadrado | Cincetti | prata, couro |

Figura 12,1 Tabela Acessórios Vendidos

Para ser atômico, o atributo Material deveria conter apenas um dos materiais e não dois ou três materiais na mesma coluna. Desdobrando o atributo Material:

| Código | Tipo | Forma | Marca | Material 1 | Material 2 | Material 3 |
|--------|---------|------------|----------|------------|------------|------------|
| 001 | brinco | losango | M&M | prata | ouro 18K | |
| 003 | anel | redonda | Stylus1 | ouro18k | prata | diamante |
| 005 | efígie | retangular | R12Z | perola | ouro18k | |
| 008 | relógio | quadrado | Cincetti | prata | couro | |

Figura 12.2 Tabela Acessórios com dados não atômicos

OK, resolvemos o problema de atributo atômico, mas note na tabela acima ainda não está na 1NF, pois veja, existem repetição de materiais nas próprias colunas de Materiais na tabela de acessórios e isso acontece porque cada tipo de acessório pode estar formado por mais de um material.

Aplicando a regra na 1FN, devemos retirar da tabela todos os grupos ou itens repetitivos e formar uma nova entidade ou tabela. Também devemos transportar a

chave primária da entidade original para a nova entidade gerada.

Como resultado desta etapa, ocorre um desdobramento dos dados em duas tabelas, a saber:

| Código | Tipo | Forma | Marca |
|--------|---------|------------|----------|
| 001 | brinco | losango | M&M |
| 003 | anel | redonda | Stylus1 |
| 005 | efígie | retangular | R12Z |
| 008 | relógio | quadrado | Cincetti |

Figura 12.3 Tabela Acessórios sem material

| Código | Material |
|--------|----------|
| 001 | prata |
| 001 | ouro 18K |
| 003 | ouro18k |
| 003 | prata |
| 003 | diamante |
| 005 | perola |
| 005 | ouro 18k |
| 008 | prata |
| 008 | couro |

Figura 12.4 Tabela Materiais

Observamos as tabelas criadas acima:

- Na figura 12.3 ficamos com os dados código, tipo, forma e marca.
- Na figura 12.4, ficamos com os dados código do acessório e material.
- O código do acessório deve existir em ambas as tabelas, para poder identificar se existem associações entre as duas tabelas.

==> Agora, nenhuma dessas tabelas possui grupos repetidos e cada atributo em ambas as entidades contém um valor simples ou atômico. Uma tabela que resulta de uma divisão como essa está na 1FN.

Início da atividade

Atividade 2 - Atende ao objetivo 2

A tabela a seguir mostra os diversos produtos de informática para cada pedido.

Normalize essa tabela na 1FN:

| Número Pedido | Código Fornecedor | Nome Fornecedor | Identidade | Endereço | Data Emissão | Produtos | | | |
|---------------|-------------------|-----------------|------------|----------|--------------|----------|-------|------|-------------|
| | | | | | | Código | Nome | Qtde | Preço Unit. |
| 02 | 012342 | Casa SS | 0231441-2 | E 11 RJ | 05/04/11 | 02221 | DVD | 100 | 0,99 |
| | | | | | | 01341 | CD-R | 200 | 0,40 |
| | | | | | | 04122 | CD-RW | 500 | 0,60 |

| | | | | | | | | | |
|----|--------|-----------|-----------|--------------|----------|-------|----------|-----|-------|
| 03 | 041211 | Computer7 | 1411981-0 | Bahia 82 SP | 21/04/11 | 02711 | Pendrive | 500 | 12,99 |
| | | | | | | 07107 | Bateria | 127 | 14,99 |
| 04 | 032329 | RioPlus | 3210062-1 | Leste 231 SC | 17/05/11 | 05123 | Modem | 300 | 15,99 |
| | | | | | | 06410 | Cargador | 15 | 5,99 |

Diagramação, inserir um espaço de 10 linhas para cada item de resposta.

Resposta Comentada

Note que cada pedido está representado por uma só linha, e as colunas de Produtos são um item de repetição onde aparecem vários produtos de um mesmo pedido. No caso do número de pedido 02 existem três linhas de produtos em uma única linha de dados. Isso porque em algumas vezes é preciso processar dois ou mais produtos num pedido. Claramente, observamos que está tabela ainda não está na 1FN. Devemos:

1. Transformar os atributos em atributos atômicos: Analisando os atributos existentes podemos identificar alguns que podem merecer cuidados adicionais, Vejamos:

- O atributo Identidade será tratado de forma atômica, pois não será separado do dígito controlador.
- O atributo Endereço será desmembrado em Rua, Número e Estado, por se tratar de dados com naturezas completamente distintas.
- O atributo Data Emissão será tratado de forma atômica através domínio DATE. Este é um tipo de dado especial em que toda a data é considerada como um atributo atômico.

Obtendo assim a primeira tabela:

| Número Pedido | Código Fornecedor | Nome Fornecedor | Identidade | Rua | Número | Estado | Data Emissão | Produtos | | | |
|---------------|-------------------|-----------------|------------|-------|--------|--------|--------------|----------|----------|------|-------------|
| | | | | | | | | Código | Nome | Qtde | Preço Unit. |
| 02 | 012342 | Casa SS | 0231441-2 | E | 11 | RJ | 05/04/11 | 02221 | DVD | 100 | 0,99 |
| | | | | | | | | 01341 | CD-R | 200 | 0,40 |
| | | | | | | | | 04122 | CD-RW | 500 | 0,60 |
| 03 | 041211 | Computer7 | 1411981-0 | Bahia | 82 | SP | 21/04/11 | 02711 | Pendrive | 500 | 12,99 |
| | | | | | | | | 07107 | Bateria | 127 | 14,99 |
| 04 | 032329 | RioPlus | 3210062-1 | Leste | 231 | SC | 17/05/11 | 05123 | Modem | 300 | 15,99 |
| | | | | | | | | 06410 | Cargador | 15 | 5,99 |

2. Extrair com os grupos de dados repetidos: Seguido pela explicação dada acima, devemos retirar da tabela todos os grupos ou itens repetitivos e com eles dar origem a novas linhas da tabela onde o conteúdo das demais colunas será o mesmo da linha original. Também devemos transportar a chave primária da entidade original para a nova entidade gerada. Fazemos isso e obtemos as seguintes tabelas:

| Número Pedido | Código Fornecedor | Nome Fornecedor | Identidade | Rua | Número | Estado | Data Emissão |
|---------------|-------------------|-----------------|------------|-------|--------|--------|--------------|
| 02 | 012342 | Casa SS | 0231441-2 | E | 11 | RJ | 05/04/11 |
| 03 | 041211 | Computer7 | 1411981-0 | Bahia | 82 | SP | 21/04/11 |
| 04 | 032329 | RioPlus | 3210062-1 | Leste | 231 | SC | 17/05/11 |

| Número Pedido | Código Produto | Nome Produto | Qtde | Preço Unit. |
|---------------|----------------|--------------|------|-------------|
| 02 | 02221 | DVD | 100 | 0,99 |
| 02 | 01341 | CD-R | 200 | 0,40 |
| 02 | 04122 | CD-RW | 500 | 0,60 |
| 03 | 02711 | Pendrive | 500 | 12,99 |
| 03 | 07107 | Bateria | 127 | 14,99 |
| 04 | 05123 | Modem | 300 | 15,99 |
| 04 | 06410 | Cargador | 15 | 5,99 |

Observamos as tabelas criadas acima:

- Utiliza-se o campo Número de Pedido como atributo chave junto com o código de produto, pois não poderá haver o mesmo código de produto e número de pedido, portanto, trata-se de uma chave composta.

==> Nenhuma dessas tabelas possui grupos repetidos, cada atributo em ambas as entidades contém um valor simples ou atômico, então as tabelas acima estão na 1FN.

Fim da atividade

O intuito da 2FN é avançar, ainda mais, na direção de obter tabela(s) que não possua anomalias.

Segunda Forma Normal (2FN)

Regra 2FN: Além de atingir a 1NF, todos os atributos que não são chaves não dependem parcialmente dessa chave.

Vejamos:

- Para que se aplique a regra da 2FN sobre uma tabela deve-se garantir, primeiramente, que ela esteja na 1FN. Caso contrário, não haverá sentido em se aplicar a 2FN.
- "...todos os atributos que não são chaves", este texto deixa claro que devemos "excluir" de nossa análise as colunas formadoras da chave primária dessa entidade.
- "...não dependem parcialmente dessa chave", deve-se perguntar a cada atributo, que não seja a chave, se ele depende apenas da chave da entidade. Caso contrário, devemos separar os atributos independentes e criar dentre os atributos separados uma nova chave para esta nova entidade. Essa chave deve ser mantida na entidade original como atributo de relacionamento entre ambas as entidades. Desta forma, não se perde qualquer informação no modelo.

Exemplo:

Imagine uma tabela chamada Prova_Vestibular:

| Código Curso | Código Aluno | Data de inscrição | Número da sala | Nome do Aluno |
|--------------|--------------|-------------------|----------------|---------------|
| 0123 | 01211131 | 03/05/2011 | 101 | Lucas Santos |
| 0222 | 03121457 | 02/05/2011 | 103 | Maria Silva |
| 0311 | 03210001 | 25/04/2011 | 101 | Jonas Pereira |
| 0311 | 15663011 | 30/04/2011 | 105 | Sônia Castro |
| 0425 | 01211131 | 04/03/2011 | 110 | Lucas Santos |

Figura 12.5 Tabela Prova-vestibular

Considere-se o código do curso e o código do aluno como chave primária, pois em alguns casos vários alunos realizaram a prova de vestibular para o mesmo curso e em outros casos um aluno realiza a inscrição para diferentes cursos; é claro que isto é possível se as provas de vestibular forem realizadas em datas diferentes.

Observamos que:

- A tabela já está na 1FN, mas ainda não, na segunda forma normal.
- Existe atributos que possuem dependência somente parte da chave. Ex:
 - ➔ Conhecendo o código do aluno podemos determinar seu nome mesmo que não soubemos o código do curso. O mesmo é válido para o número de sala onde será aplicado a prova.

→ Conhecendo o código do curso podemos também determinar o número da sala ainda que não saibamos o código do aluno.

Então tanto o número da sala e o nome do aluno mostram a dependência parcial da chave primária. Elas não participam da chave primária da entidade Prova_Vestibular. Devemos criar novas entidades e excluí-las da entidade original.

- Existe atributos que possuem dependência total da chave. Ex:

→ Para determinar quando foi realizada a data de inscrição da prova devemos conhecer tanto o código do aluno e o código do curso, pois se tivermos o código do aluno podemos obter diversas datas de inscrição.

→ De igual forma, se conhecermos o código do curso podemos obter diferentes datas de inscrição, uma para cada curso onde o aluno se inscreveu.

Aplicando a regra 2FN visto as dependências parciais obtemos as seguintes tabelas:

| Código Curso | Código Aluno | Data de inscrição |
|--------------|--------------|-------------------|
| 0123 | 01211131 | 03/05/2011 |
| 0222 | 03121457 | 02/05/2011 |
| 0311 | 03210001 | 25/04/2011 |
| 0311 | 15663011 | 30/04/2011 |
| 0425 | 01211131 | 04/03/2011 |

Figura 12.6 Tabela Inscrição-Vestibular

| Código Aluno | Nome do Aluno |
|--------------|---------------|
| 01211131 | Lucas Santos |
| 03121457 | Maria Silva |
| 03210001 | Jonas Pereira |
| 15663011 | Sônia Castro |
| 01211131 | Lucas Santos |

Figura 12.7 Tabela Aluno-Vestibular

| Código Curso | Número da sala |
|--------------|----------------|
| 0123 | 101 |
| 0222 | 103 |
| 0311 | 101 |
| 0311 | 105 |
| 0425 | 110 |

Figura 12.8 Tabela Curso-Sala

Note que obtivemos três tabelas distintas que surgiram após da normalização 2FN.

Início da atividade

Atividade 3 - Atende ao objetivo 3

A seguir normalize a tabela Pedido-Eletrodoméstico na 2FN. Nesta tabela temos diversos eletrodomésticos e seus correspondentes números de pedidos, data de emissão, nome e identidade da empresa fornecedora, endereço, código, nome, quantidade e preço unitário de cada produto.

Tabela Pedido-Eletrodoméstico

| Código Pedido | Código Produto | Data Emissão Pedido | Nome Empresa Fornecedor | CNPJ | Rua | Número | Estado | Nome Produto | Qtde | Preço Unit. |
|---------------|----------------|---------------------|-------------------------|-------------|------|--------|--------|----------------|------|-------------|
| 11 | 191 | 11/02/11 | Casas Ricardo e Ana | 02314418992 | Mira | 11 | SP | Fogão | 6 | 450,99 |
| 11 | 022 | 11/02/11 | Casas Ricardo e Ana | 02314418992 | Flor | 231 | RJ | Ferro | 30 | 32,00 |
| 13 | 083 | 20/03/11 | Ponto Quente | 03210062121 | Fé | 21 | RJ | Lavadora | 10 | 890,99 |
| 14 | 111 | 07/04/11 | Loja Paulão | 02121212123 | Kan | 565 | SP | Liquidificador | 21 | 79,99 |
| 14 | 156 | 07/04/11 | Loja Paulão | 02121212123 | Lua | 1290 | SP | Geladeira | 12 | 1200 |
| 15 | 112 | 09/05/11 | Ponto Azul | 09121212122 | Ana | 343 | SC | Batedora | 30 | 89,99 |
| 16 | 078 | 09/05/11 | Casa Bahia123 | 01411981000 | Fogo | 103 | AL | TV | 20 | 2100 |

Diagramação, inserir um espaço de 10 linhas para cada item de resposta.

Resposta Comentada

Aplicando-se a regra 2FN sobre a tabela Pedido-Eletrodoméstico, deveremos:

- A tabela já está na 1FN.
- A chave primária estará composta pelos atributos código do pedido e o código do produto.
- Analisar os atributos não chaves com dependência na chave primária:
 - ✗ Independente do código produto, conhecendo o código do pedido, podemos determinar a data emissão, nome da empresa fornecedora, CNPJ, rua, número e estado. Cada pedido diferente desses atributos podem ser determinados.
 - ✗ Conhecendo o código do produto podemos determinar o nome do produto independente ao código do pedido.

- Para se conhecer as quantidades que foram pedidas à empresa fornecedora é necessário saber qual foi o código do pedido e o código do produto. Considerando as informações do código do pedido teremos várias quantidades, uma para cada produto desse pedido. Se considerarmos o código do produto teremos várias quantidades, uma para cada pedido onde apareça esse produto. Neste caso mostra uma dependência total.
- Se o atributo preço unitário for associado ao produto, esse atributo possuirá dependência parcial para a chave código de produto.

Criando novas entidades com os atributos dependentes parcialmente das chaves e excluindo esses atributos na tabela original Pedido-Eletrodoméstico, obteremos as seguintes tabelas:

1. Atributos dependentes totalmente da chave

| Código Pedido | Código Produto | Qtde | Preço Unit. |
|---------------|----------------|------|-------------|
| 11 | 191 | 6 | 450,99 |
| 11 | 022 | 30 | 32,00 |
| 13 | 083 | 10 | 890,99 |
| 14 | 111 | 21 | 79,99 |
| 14 | 156 | 12 | 1200 |
| 15 | 112 | 30 | 89,99 |
| 16 | 078 | 20 | 2100 |

2. Atributos dependentes parcialmente da chave código de pedido:

| Código Pedido | Data Emissão Pedido | Nome Empresa Fornecedor | CNPJ | Rua | Número | Estado |
|---------------|---------------------|-------------------------|-------------|------|--------|--------|
| 11 | 11/02/11 | Casas Ricardo e Ana | 02314418992 | Mira | 11 | SP |
| 11 | 11/02/11 | Casas Ricardo e Ana | 02314418992 | Flor | 231 | RJ |
| 13 | 20/03/11 | Ponto Quente | 03210062121 | Fé | 21 | RJ |
| 14 | 07/04/11 | Loja Paulão | 02121212123 | Kan | 565 | SP |
| 14 | 07/04/11 | Loja Paulão | 02121212123 | Lua | 1290 | SP |
| 15 | 09/05/11 | Ponto Azul | 09121212122 | Ana | 343 | SC |
| 16 | 09/05/11 | Casa Bahia123 | 01411981000 | Fogo | 103 | AL |

3. Atributo nome de produto dependente parcialmente da chave código de produto.

| Código Produto | Nome Produto |
|----------------|----------------|
| 191 | Fogão |
| 022 | Ferro |
| 083 | Lavadora |
| 111 | Liquidificador |
| 156 | Geladeira |
| 112 | Batedora |
| 078 | TV |

Assim, a tabela Pedido-Eletrodoméstico foi normalizada 2FN, obtendo três tabelas.

Fim da atividade

O processo de normalização da 3FN também dará origem a novas tabelas; este processo garante uma maior fidelidade ao processo de normalização, completando a 2FN.

Terceira Forma Normal (3FN)

A forma 3FN estende a forma 2FN para incluir a eliminação de dependências transitivas. A Dependência Transitiva surge quando no momento em que um atributo está dependendo de outro atributo que depende da chave primária. Exemplo:

Atributo A1 depende do Atributo B1;

Atributo B1 depende do Atributo C1 compondo a chave primária;

Logo, existe uma dependência transitiva do Atributo A1 em relação ao Atributo C1.

Regra 3FN: A tabela deve estar na 2FN e se nenhum atributo não pertencente à chave fica determinada transitivamente por esta.

Vejamos:

- Para que se aplique a regra da 3FN deve garantir os passos anteriores (2FN e 1FN). Sem eles não haverá sentido aplicar a 3FN.
- "...se nenhum atributo não pertencente à chave...", este texto sugere que deverão ser analisados os atributos não pertencentes à chave.
- "...fica determinada transitivamente por esta...", a dependência transitiva de

uma chave, só será possível se a tabela tiver ao menos duas colunas (ou atributos) não pertencentes à chave. Se a tabela possui uma só coluna que não pertence à chave primária, então essa tabela já está na 3FN.

- Para os atributos dependentes transitivamente da chave:
 - ✗ Criar novas entidades onde a chave primária será(ão) a(s) coluna(s) que determinou(aram) o valor da coluna analisada. Agregar a essas entidades as colunas dependentes transitivamente.
 - ✗ Excluir da entidade original as colunas dependentes transitivamente das chaves mantendo, porém, a coluna determinante da transitividade na tabela.

Exemplo:

Imagine que temos a tabela de compras dos clientes de uma loja de um Shopping e a tabela contendo os nomes dos clientes das lojas.

| Código Cliente | Preço Unitário | Qtes itens | Data última compra | Nome Cidade | Tipo Cidade |
|----------------|----------------|------------|--------------------|-------------|-------------|
| 03413 | 31,99 | 14 | 01/12/2009 | nome1 | interior |
| 07342 | 21,99 | 15 | 04/03/2011 | nome2 | capital |
| 01212 | 47,99 | 20 | 12/05/2011 | nome3 | interior |
| 04538 | 99,99 | 12 | 1/04/2011 | nome4 | metrópole |
| 07342 | 12,99 | 28 | 1/04/2011 | nome5 | capital |

Figura 12.9 Tabela Compra-Cliente

| Código Cliente | Nome Cliente |
|----------------|--------------|
| 03413 | Ana Pereira |
| 01212 | Karla Costa |
| 04538 | João Santos |
| 07342 | Maria Hagin |

Figura 12.10 Tabela Cliente

- A Tabela Compra-Cliente, os atributos preço unitário (descontos especiais por clientes), quantidades de itens comprados, data da última compra, município e o tipo de cidade onde reside possuem dependência total com a chaves primária (código de cliente). Todo cliente é avaliado em função desses atributos.

- O tipo de cidade pode ser determinado através do nome da cidade. O nome da cidade é determinado pelo código do cliente (chave primária), então existe uma dependência transitiva para o atributo tipo de cidade em relação ao atributo código do cliente.
- Para o atributo tipo de cidade, que é dependente transitivamente da chave, devemos criar uma nova entidade em que a chave primária será o atributo que determinou o valor do atributo analisado (nome da cidade), agregando a essa entidade o atributo tipo de cidade.
- Devemos excluir da tabela origem (Compra-Cliente) o atributo tipo de cidade mantendo o atributo (nome da cidade) determinante na transitividade na tabela.
- A Tabela Clientes, possui uma só coluna que não pertence à chave primaria; então essa tabela já está na 3FN.

Aplicando a regra 3FN, obtemos as seguintes tabelas:

| Código Cliente | Preço Unitário | Qtes itens | Data última compra | Nome Cidade |
|----------------|----------------|------------|--------------------|-------------|
| 03413 | 31,99 | 14 | 01/12/2009 | nome1 |
| 07342 | 21,99 | 15 | 04/03/2011 | nome2 |
| 01212 | 47,99 | 20 | 12/05/2011 | nome3 |
| 04538 | 99,99 | 12 | 1/04/2011 | nome4 |
| 07342 | 12,99 | 28 | 1/04/2011 | nome5 |

Figura 12.11 Tabela Compra-Cliente Modificada

| N o c i d a d e | Código Cliente | Nome Cliente |
|--------------------------------------|-------------------|-----------------|
| m e Ci | 03413 | Ana Pereira |
| C i da de | 01212 | Karla Costa |
| d a d e | 04538 | João Santos |
| n o te m ri | 07342 | Maria Hagin |

Início da atividade

Atividade 4 - Atende ao objetivo 4

A seguir normalize a tabela Produção na 3FN.

| Código | Nome | Tempo Produção(horas) | Código Chefe | Nome Chefe |
|--------|-----------|-----------------------|--------------|------------|
| 0123 | Matéria 1 | 100 | 0017 | Carlos |
| 3431 | Matéria 2 | 150 | 0459 | Pedro |
| 5486 | Matéria 3 | 80 | 0321 | Augusto |
| 3421 | Matéria 4 | 220 | 9651 | Ana |

Diagramação, inserir um espaço de 6 linhas para cada item de resposta.

Resposta Comentada

- A Tabela Produção, os atributos código, nome, tempo de produção, código do chefe e o nome do chefe possuem dependência total com a chave primária código da matéria prima.
- O nome do chefe pode ser determinado através do código do mesmo. O código do chefe é determinado pelo código da matéria-prima, então existe uma dependência transitiva para o atributo nome do chefe em relação ao atributo código da matéria prima.
- Para o atributo nome do chefe, que é dependente transitivamente da chave, devemos criar uma nova entidade em que a chave primária será o atributo que determinou o valor do atributo analisado (código do chefe), agregando a essa entidade o atributo nome do chefe.

Aplicando a regra 3FN, obtemos as seguintes tabelas:

| Código | Nome | Tempo Produção(horas) | Código Chefe |
|--------|-----------|-----------------------|--------------|
| 0123 | Matéria 1 | 100 | 0017 |
| 3431 | Matéria 2 | 150 | 0459 |
| 5486 | Matéria 3 | 80 | 0321 |
| 3421 | Matéria 4 | 220 | 9651 |

e

| Código Chefe | Nome Chefe |
|--------------|------------|
| 0017 | Carlos |
| 0459 | Pedro |

| | |
|------|---------|
| 0321 | Augusto |
| 9651 | Ana |

Fim da atividade

A Terceira Forma Normal Estendida foi desenvolvida em 1974 para tratar certos tipos de anomalias não tratadas pela forma 3NF originalmente definida.

Terceira Forma Normal Estendida (BCNF)

É uma forma normal usada em normalização de banco de dados que consiste em uma versão mais rigorosa da terceira forma normal (3FN). Uma entidade atinge a forma normal de Boyce-Codd se e somente se, para cada um de suas dependências funcionais não triviais $X \rightarrow Y$, X é uma super-chave de Y, ou seja, X é a chave candidata.

Considerações:

- Uma tabela na forma 3FN que não tem muitas chaves candidatas sobrepostas é garantida estar na forma BCNF.
- Dependendo de quais são suas dependências funcionais, uma tabela na forma 3FN com duas ou mais chaves candidatas sobrepostas pode ou não estar na forma BCNF.
- Uma super-chave é um conjunto de atributos que identifica univocamente uma relação. Em outras palavras não podem existir duas ou mais linhas da tabela com o(s) mesmo(s) valores de uma super-chave.
- Uma chave candidata é uma super-chave minimal ou seja, qualquer sub-conjunto dela não é super-chave.
- Uma Super-Chave de uma Tabela não pode conter Valor Nulo (**NULL**).
- Na prática, uma tabela está em *BCNF* se estiver em *3NF* e **não existir** dependência funcional dentro da chave primária, ou seja, se todos os atributos são funcionalmente dependentes da chave, de toda a chave e nada mais do que a chave. Ou, em outras palavras, todos os determinantes são chaves candidatas.

Exemplo:

Considere uma tabela que representa uma reserva de diversas quadras para eventos

desportivos:

| Nº quadra | Inicio | Término | Categoria Reserva |
|-----------|----------|----------|-------------------|
| 1 | 08:00:00 | 09:30:00 | A |
| 1 | 10:30:00 | 12:15:00 | A |
| 1 | 14:00:00 | 15:30:00 | B |
| 2 | 11:00:00 | 12:30:00 | C |
| 2 | 14:00:00 | 15:30:00 | C |
| 3 | 18:00:00 | 19:30:00 | E |

Figura 12.13 Tabela Reserva de quadras

- Cada linha da tabela representa uma reserva da quadra.
- Cada reserva possui um tipo de categoria associada a ela.
- Existem três tipos de categorias.
- As chaves candidatas são:
 - num_quadra e inicio
 - num_quadra e término
 - categoria e inicio
 - categoria e término
- Esta tabela está na 3FN, pois ela atingiu a 2FN e nenhum atributo que não pertencente à chave fica determinada transitivamente por esta.
- A tabela Reserva de quadras não está na forma BCNF e isso ocorre pela dependência categoria → num_quadra e num_quadra não é uma chave.
- O atributo determinante (categoria) não é uma chave candidata nem um super conjunto de uma chave candidata.

Vejamos:

- Não existe nada que nos impeça de associar uma categoria a uma reserva de quadra 1 ou 2.
- Trata-se de um problema claro de contradição, pois uma categoria só deveria ser aplicada a uma única quadra.

A tabela pode ser alterada da seguinte forma:

| Categoria | Nº quadra | Status Membro | Nº quadra | Inicio | Término | Status Membro |
|-------------------------------|-----------|---------------|-----------|----------|----------|---------------|
| A | 1 | Sim | 1 | 08:00:00 | 09:30:00 | Sim |
| B | 1 | Não | 1 | 10:30:00 | 12:15:00 | Sim |
| C | 2 | Sim | 1 | 14:00:00 | 15:30:00 | Não |
| D | 3 | Não | 2 | 11:00:00 | 12:30:00 | Sim |
| Figura 12.14 Tabela Categoria | | | 2 | 14:00:00 | 15:30:00 | Sim |
| | | | 3 | 18:00:00 | 19:30:00 | Não |

- As chaves candidatas para a tabela Categoria são:
 - * categoria e (num_quadra, membro).
- As chaves candidatas para a tabela Reserva de quadras modificada são:
 - * (num_quadra, inicio) e (num_quadra, termino).

Obtivemos uma categoria associada com duas quadras diferentes, então o efeito anormal afetando a tabela original foi eliminado. Finalmente, ambas as tabelas estão na forma BCNF.

Início da atividade

Atividade 5 - Atende ao objetivo 5

Dado o esquema relacional: (id_Cliente, num_empréstimo, quantia). Analise se o esquema está na sua forma BCNF.

Diagramação, inserir um espaço de 6 linhas para cada item de resposta.

Resposta Comentada

O esquema não está na BCNF. Verifica-se a dependência funcional no atributo num_Empréstimo → quantia. No entanto Empréstimo não é uma super chave.

Se R um esquema que não está na BCNF, então existe pelo menos uma dependência funcional não trivial X→ Y tal que X não é super chave para R.

A tabela pode ser alterada da seguinte forma:

Empréstimo, quantia) e (idCliente, Empréstimo)

Fim da atividade

Inicio da atividade online

Atividade 1 - Atende ao objetivo 2

Acesse o ambiente virtual e resolva esta atividade, enviando sua resposta ao tutor. No contexto de um sistema de controle acadêmico, considera a tabela abaixo:

Matricula

([CodAluno](#),[CodTurma](#), [CodDisciplina](#), [NomeDisciplina](#), [NomeAluno](#),
[CodLocalNascAluno](#), [NomeLocalNascAluno](#), [Telefone](#))

As colunas possuem o seguinte significado:

[CodAluno](#) - código do aluno matriculado.

[CodTurma](#) - código da turma na qual o aluno está matriculado (código é o identificador de turma).

[CodDisciplina](#) - código que identifica a disciplina da turma.

[NomeDisciplina](#) - nome de uma disciplina da turma.

[NomeAluno](#) - nome do aluno matriculado.

[CodLocalNascAluno](#) - código da localidade em que nasceu o aluno.

[NomeLocalNascAluno](#) - nome da localidade em que nasceu o aluno.

[Telefone](#) – telefone do aluno matriculado.

Verifique se a tabela obedece a 1FN, 2FN e a 3FN. Caso não obedeça, faça as transformações necessárias.

Fim da Atividade Online

Conclusão

Todo banco de dados relacional deve ser projetado para atingir requisitos de qualidade, desempenho e escalabilidade dos dados. A eficiência devem ser mantidos num estado consistente e lógico. A normalização apoia a definição de requisitos de projeto de modelo de dados removendo as falhas de consistências de dados e anomalias de manipulação de dados, preservando os elementos de um sistema de alto desempenho.

Próxima Aula

Todas as regras de negocio de uma empresa são realizadas, isto é centralizadas e executadas no servidor de banco de dados, por isso quando é realizado um evento ou uma ação na aplicação seja ela desktop ou web os comandos são enviados ao servidor de banco de dados, que vai interpretá-los, executá-los retornando apenas a resposta ao que foi solicitado, reduzindo assim a tráfego na rede, pois as informações já estarão resumidas, ou seja, selecionadas de acordo com o solicitado pela aplicação. Isso será visto com mais detalhes na próxima aula no tema de Procedures e Triggers.

Resumo

Normalização:

- É o processo formal passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros.
- Normalização de Dados consiste em definir o formato lógico adequado para as estruturas de dados identificados no projeto lógico do sistema, com o objetivo de minimizar o espaço utilizado pelos dados e garantir a integridade e confiabilidade das informações.

Benefícios:

- Estabilidade do Modelo Lógico: capacidade de um modelo manter-se inalterado face a mudanças que venham acontecer.
- Flexibilidade: aumenta o grau de adaptabilidade do banco de dados em função de mudanças no Mundo Real (Regras de Negócio e Requisitos de Informação)
- Integridade: elimina as redundâncias dos dados na tabela.
- Economia: minimiza os custos de qualquer esforço, tempo e armazenamento.
- Expressividade: aumenta o grau de consistência das informações armazenadas no banco de dados.

Formas Normais

- A normalização é feita através da análise dos dados que compõem as estruturas utilizando o conceito chamado “Formas Normais (FN)”. As FN são conjuntos de restrições nas quais os dados devem satisfazê-las.
- A normalização completa dos dados é feita seguindo as restrições das três

formas normais existentes, sendo que a passagem de uma FN para outra é feita tendo como base o resultado obtido na etapa anterior, ou seja, na FN anterior.

Primeira Forma Normal (1FN)

Regra 1FN: Cada atributo de uma tabela deve ter apenas um único valor.

Devemos,

1. transformar os atributos em atributos atômicos;
2. extrair com os grupos de dados repetidos.

Segunda Forma Normal (2FN)

Regra 2FN: Além de atingir a 1NF, todos os atributos que não são chaves não dependem parcialmente dessa chave.

Terceira Forma Normal (3FN)

A forma 3FN estende a forma 2FN para incluir a eliminação de dependências transitivas. A Dependência Transitiva surge quando no momento em que um atributo está dependendo de outro atributo que depende da chave primária. Exemplo:

Atributo A1 depende do Atributo B1;

Atributo B1 depende do Atributo C1 compondo a chave primária;

Logo, existe uma dependência transitiva do Atributo A1 em relação ao Atributo C1.

Regra 3FN: A tabela deve estar na 2FN e se nenhum atributo não pertencente à chave fica determinada transitivamente por esta.

Terceira Forma Normal Estendida (BCNF)

É uma forma normal usada em normalização de banco de dados que consiste em uma versão mais rigorosa da terceira forma normal (3FN). Uma entidade atinge a forma normal de Boyce-Codd se e somente se, para cada um de suas dependências funcionais não triviais $X \rightarrow Y$, X é uma super-chave de Y, ou seja, X é a chave candidata.

Referências Bibliográficas

CARVALHO C.R., 2006. SQL-Guia Prático. 2ed. Rio de Janeiro. Brasport.

DATE C.J., 2003. Introdução a Sistemas de Bancos de Dados. 8ed. Americana. Rio de Janeiro. Elsevier.

ELMASRI R., Navathe S., 2009. Sistemas de Banco de Dados. 5ed. São Paulo. Pearson Addison Wesley.

- HEUSER C.A. 2009. Projeto de Banco de Dados. 4ed. Porto Alegre. Bookman.
- SETZER V.W. & Corrêa da Silva F.S. 2005. Bancos de Dados. 1ed. São Paulo. Edgard Blucher.
- SILBERSCHATZ A., Korth H., 2008. Sistema de Banco de Dados. 3ed. São Paulo. Pearson.

Curso de Extensão: Modelando e Implementando Bancos de Dados Relacionais
Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 13

Stored Procedures e Triggers

Meta

Apresentar os comandos importantes para à criação e manipulação de Stored Procedures e Triggers (gatilhos).

Objetivos

Ao final desta aula, esperamos que você seja capaz de:

1. Elaborar consultas utilizando Stored Procedures:
 - ▲ Definição.
 - ▲ Criação.
 - ▲ Alteração.
 - ▲ Exclusão.
2. Utilizar o conceito de CURSOR.
3. Aplicar consultas utilizando Triggers:
 - ▲ Eventos triggers: INSERT, UPDATE ou DELETE;
 - ▲ Exclusão da triggers.

Pré-requisitos

Para se ter um bom aproveitamento desta aula, é importante você relembrar os comandos DDL e DML aprendidos nas aulas 8, 9,e 10 respetivamente.

Introdução

Porque ele faz o que um gatilho de uma arma faz!

Você já deve ter ouvido o termo Stored Procedure; isto, se não trabalhou com algumas delas! Stored procedures são rotinas gravadas dentro de um SGBD; são como um pedaço de código de programa que fica armazenado para ser posteriormente utilizado, facilitando, assim, a vida dos administradores do sistema ou usuários. A utilização de Stored Procedures é uma técnica eficiente que executa operações repetitivas. Assim, ao invés de se digitar os comandos cada vez que determinada operação necessite ser executada, criamos um Stored Procedures e o chamamos. Imaginemos que estamos criando uma aplicações em Delphi e que a mesma utilize um banco de dados do SQL. Sem a utilização de Stored Procedures, a aplicação deveria enviar um conjunto de comandos em SQL necessários à execução de cada tarefa. Imagine também ter que enviar os comandos em várias partes do programa e qualquer modificação demandaria uma revisão em todas as partes do programa que enviam estes comandos.

Uma solução para o problema acima seria isolar e fechar os comandos SQL em uma função e chamá-la só quando for necessário. Triggers (ou gatilhos) é um tipo de Stored Procedure que possui mecanismos para ser executado de forma automática, e por isso é chamada de “gatilho”. A sua execução vai depender de um evento no banco de dados, ou seja, um comando SQL executado em uma determinada tabela.

Agora que já conhecemos as vantagens da utilização das Stored Procedures e os Triggers , vamos à prática!

Fim da Introdução

Stored Procedure.

O que é?

Stored Procedure ou procedimentos armazenados são uma coleção de comando em SQL, compilados e armazenados no servidor. Geralmente eles representam tarefas repetitivas e possuem parâmetros de entrada ou saída. A Stored Procedure serve para:

- ▲ melhorar o tráfego na rede,
- ▲ melhorar a performance das aplicações,
- ▲ criar mecanismos de segurança e
- ▲ melhorar a manutenção dos códigos SQL e das aplicações que acessam o

banco de dados.

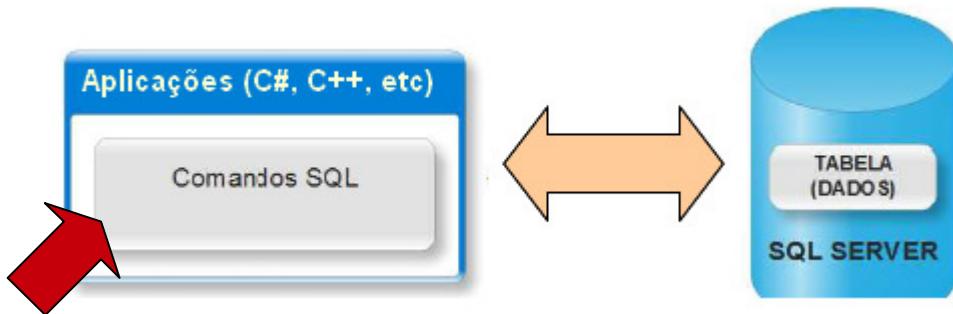


Figura 13.1:Modelo de Acesso ao Banco de Dados sem utilização de Stored Procedures

Ao executar a procedure pela 1ºvez ela é compilada e a cada execução seus resultados são colocados em cache; o cache é semelhante a uma memória que guarda as últimas operações do banco de dados, caso a procedure seja executada novamente ela pega o resultado do cache diminuindo o esforço do banco de dados e aumentando a velocidade de acesso.

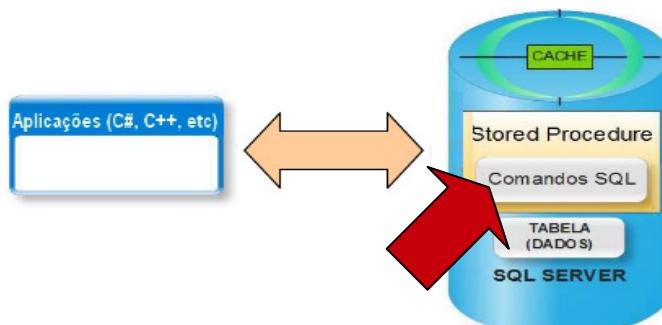


Figura 13.2 Modelo de Acesso ao Banco de Dados utilizando Stored Procedures

Quando utilizar procedures?

- ▲ Quando temos várias aplicações escritas em diferentes linguagens ou rodam em plataformas diferentes, porém todas elas executam a mesma função.
- ▲ Quando damos prioridade à consistência e segurança.

Exemplo: Os bancos (Itaú, Bradesco, Real, etc), em geral, utilizam Stored Procedures para todas as operações em comum. Os procedimentos podem assegurar que as operações sejam registradas de forma correta e segura.

Por que é mais seguro?

Seguindo a linha de raciocínio dos bancos, utilizando Stored Procedures outras aplicações e usuários não conseguiriam nenhum tipo de acesso às tabelas do banco de dados de forma direta. Eles poderiam apenas executar as Stored Procedures, que rodam ações específicas e determinadas pelos DBAs e desenvolvedores.

Como criar a Stored Procedure?

Antes de criar esta rotina, é necessário mudar o delimitador de comandos do MySQL, pois no conteúdo da rotina existe o caractere “ ; ”, que não pode ser confundido pelo sistema como uma finalização de comando. Não esqueça de executar o delimitador: Delimiter //.

Vejamos a sintaxe de criação da stored procedure:

CREATE PROCEDURE <banco de dados>.<nome da rotina>

(Parâmetros de entrada e saída) <nome da variável> <tipo de dados>
BEGIN
<comandos Select into>
END:

onde:

- ▲ CREATE PROCEDURE <banco de dados>.<nome da rotina>: cria a stored procedure de nome da rotina no banco de dados que já existe.
- ▲ (Parâmetros de entrada e saída) <nome da variável> <tipo de dados>: o parâmetro de entrada IN, o parâmetro de saída OUT e INOUT as duas coisas. Nome da variável criada e o tipo de dados que corresponde a variável criada.
- ▲ BEGIN ... END: estrutura da stored procedure que contém os comandos a serem executados. Aqui no meio vai as instruções SQL.
- ▲ Pode-se utilizar os comandos DECLARE (declaração de variáveis locais) e SET (atribuição de valores). Permite a execução de instruções if-then-else, loops, etc; e comandos de select, insert, delete e update podem ser utilizados dentro de uma procedure

Podemos voltar o delimitador para o padrão. Agora que a rotina já está criada e incluída no banco de dados, para executá-la e armazenar os valores em variáveis, basta utilizar o comando CALL.

[Saiba +](#)

Há 5 Procedimentos (Procedures) básicos que podemos criar:

- ▲ Procedimentos Locais - São criados a partir de um banco de dados do próprio usuário;
- ▲ Procedimentos Temporários - Existem dois tipos de procedimentos temporários: **Locais**, que devem começar com # e **Globais**, que devem começar com ##;
- ▲ Procedimentos de Sistema - Armazenados no banco de dados padrão do SQL Server (Master), podemos identificá-los com as siglas SP. que se origina de stored procedure.
- ▲ Procedimentos Remotos - Podemos usar Queries Distribuídas para tais procedures. São utilizadas apenas para compatibilidade.
- ▲ Procedimentos Estendidos - Diferente dos procedimentos já citados, este tipo de procedimento recebe a extensão .dll e são executadas fora do SGBD SQL Server. São identificadas com o prefixo XP.

Fim saiba +

Como alterar uma Stored Procedure?

ALTER PROCEDURE, este comando pode ser usado para renomear uma stored procedure e para alterar suas características. Basta acessar a sua base de dados e criar uma nova query e digitar os comandos responsáveis pela alteração e executá-los.

Sintaxe:

ALTER PROCEDURE <nome da rotina> [características ...]

características:

```
NAME novo nome  
| SQL SECURITY {DEFINER | INVOKER}  
| COMMENT 'string'
```

onde:

- ▲ A característica SQL SECURITY:
 - ▲ SECURITY INVOKER indica que a stored procedure deve ser executada com os privilégios do usuário a chamou. Este é o padrão.
 - ▲ SECURITY DEFINER especifica que a stored procedure deve ser executada com os privilégios do usuário que a criou.
- ▲ A cláusula COMMENT é uma extensão do MySQL, e pode ser usada para descrever o stored procedure. Esta informação é exibida pelas instruções

SHOW CREATE PROCEDURE.

Ex: ALTER PROCEDURE SPEmpregado SQL SECURITY DEFINER;

Acabamos de alterar a stored procedure SPEmpregado para que seja executada com os privilégios do usuário que a criou.

Como excluir uma Stored Procedure?

DROP PROCEDURE é usado para deletar uma stored procedure. Isto é, a procedure especificada é removida do servidor.

DROP PROCEDURE [IF EXISTS] <nome da rotina>

A cláusula IF EXISTS é uma extensão do MySQL. Ela previne que um erro ocorra se o procedimento não existe. Um aviso é produzido e pode ser visualizado com SHOW WARNINGS. Ex: DROP PROCEDURE IF EXISTS SPEmpregado;

Acabamos de deletar a stored procedure SPEmpregado, considerando o comando IF EXISTS, claro prevenindo se por acaso o procedimento não exista.

Exemplos:

(1) Criar uma stored procedure chamada SP_Variaveis.

```
CREATE PROCEDURE SP_Variaveis()
```

```
    BEGIN
```

```
        DECLARE x, y, z INT DEFAULT 0;
```

```
        SELECT x as "var 1", y as "var 2", z as "var3";
```

```
    END;
```

Agora, vejamos como foi feito a procedure de acima:

- ▲ CREATE PROCEDURE SP_Variaveis: cria uma stored procedure de nome SP_Variaveis.
- ▲ BEGIN ... END: inicia e finaliza a stored procedure.
- ▲ DECLARE x, y, z INT DEFAULT 0: declaração das variáveis locais x, y, z do tipo inteiro e com valores iniciais igual a 0.

- ^ SELECT x as "var 1", y as "var 2", z as "var3": substitui os nomes x, y e z pelos "var1", "var2" e "var3" respectivamente.

Depois que a rotina já foi criada e incluída no banco de dados, para executá-la basta utilizar comando CALL da seguinte forma:

```
CALL SP_Variaveis();
```

| var 1 | var 2 | var 3 |
|-------|-------|-------|
| 0 | 0 | 0 |

(2) Criar a stored procedure que irá retornar as quantidades de tuplas ou registros das tabelas MovCliente2010, ContaBancos e ContaReceberPagar do banco de dados ContasGeral:

```
CREATE PROCEDURE ContasGeral. SP_ContaBancos
(
    OUT var_MovimentoCaixa INT,
    OUT var_ContaBanco INT,
    OUT var_ContraReceberPagar INT
)
BEGIN
    SELECT COUNT(LancamentoCaixa)
        into var_MovimentoCaixa from MovCliente2010;
    SELECT COUNT(Conta) into var_ContaBanco from ContaBancos;
    SELECT COUNT(LancamentoRP)
        into var_ContraReceberPagar from ContaReceberPagar;
END;
```

Observamos que:

- ^ CREATE PROCEDURE ContasGeral. SP_ContaBancos: cria uma stored procedure de nome SP_ContaBancos no banco de dados ContasGeral;
 - ^ Foram criadas algumas variáveis de saída, determinadas pelo parâmetro OUT. Estas variáveis terão seus valores incrementados pelos comandos SELECT entre a estrutura BEGIN ... END, quando a rotina for executada;
- Ex: OUT var_ContaBanco int: define uma variável de nome var_ContaBanco do

tipo inteiro que terá seu valor incrementado pelo comando SELECT into;

▲ SELECT COUNT(LancamentoCaixa) into var_MovimentoCaixa from MovCliente2010: contabiliza os dados da coluna LancamentoCaixa, da tabela MovCliente2010, e armazena o valor na variável var_MovimentoCaixa.

Depois que a rotina já foi criada e incluída no banco de dados, para executá-la e armazenar os valores em variáveis, basta utilizar comando CALL da seguinte forma:

```
CALL ContasGeral. SP_ContaBancos  
(      @TuplasMovimento,  
      @TuplasConta,  
      @TuplasReceberpagar )
```

Após de executar este comando CALL, a procedure SP_ContaBancos será executada e as variáveis de retorno dela, definidas como OUT, serão vinculadas às variáveis definidas no comando CALL, por meio do operador @.

Agora, utilizar o comando SELECT para visualizar os dados retornados pela stored procedure SP_ContaBancos da seguinte forma:

```
SELECT  
      @TuplasMovimento,  
      @TuplasConta,  
      @TuplasReceberpagar
```

Ao executar este comando SELECT, você estará listando os valores das variáveis definidas pelo comando CALL, as quais já possuem os valores retornados pela stored procedure.

- (3) Considere diversos projetos em diversos estados e só existe uma tabela chamada tb_EstadoProjeto e desejamos inserir algumas informações. Crie a stored procedure SP_InserirEstadoP para incluir qualquer registro.

| Código Estado | Nome | Código Projeto |
|---------------|-------------|----------------|
| 01 | Mendes | 0123 |
| 02 | Barra Mansa | 0111 |

| | | |
|----|---------------|------|
| 03 | Macaé | 0324 |
| 04 | Areal | 0211 |
| 05 | Nova Friburgo | 0153 |

Figura 13.3 Tabela tb_EstadoProjeto

```
CREATE PROCEDURE tb_EstadoProjeto.SP_InserirEstadoP
```

```
(      IN var_id_estado INT,
      var_nome VARCHAR(30),
      var_pcodigo INT
)
BEGIN
    INSERT INTO tb_EstadoProjeto VALUES (var_id_estado, var_nome,
      var_pcodigo);
END;
```

Vamos chamar a stored procedure que foi criada e inserir um código e nome de estado e código de projeto correspondente:

```
CALL SP_InserirEstadoP( 06, "Magé", 02101);
```

Pronto, código 06, nome do Estado Magé e código do projetos foram inseridos nas tabela Tabela tb_EstadoProjeto.

4) Utilizando a tabela CLIENTES, crie a stored procedure chamada SP_Lista_Clientes. A procedure avalia quando uma opção é 0, 1 ou qualquer outro valor. Se a opção é o valor 0, deverá mostrar nome e endereço de todos os clientes de sexo feminino, se a opção é o valor 1 deverá mostrar o código e nome de todos os clientes de sexo masculino. Qualquer outro valor de opção diferente de 0 e 1 mostrar todas as colunas da tabela CLIENTES.

```
CREATE PROCEDURE SP_Lista_Clientes
```

```
(IN opcao INT)
```

```
BEGIN
```

```
CASE opcao
```

```

WHEN 0 THEN select nome, endereço from CLIENTES where sexo = 'F';
WHEN 1 THEN select codigo, nome from CLIENTES where sexo = 'M';
else
    select * from CLIENTE;
END CASE;
END;

```

Início da atividade

Atividade 1 - Atende ao objetivo 1

Imagine um sistema de vendas e o processamento de uma venda; seja o caso de uso Processar Venda(fluxo normal):

- ▲ O usuário inicia a venda e para cada item, identifica-o e registra a quantidade;
- ▲ O sistema registra o item de venda e apresenta sua descrição como preço e total parcial;

Ao final, o caixa registra a venda. O sistema processa a venda, atualiza o status da mesma para finalizar e atualizar as informações correspondentes ao setor financeiro. Portanto crie a stored procedure para processar a Venda, o procedimento pode ser definido como:

- a) Nome : SP_ProcessaVendas
- b) Parâmetro de entrada: codigo_compra e código_venda.
- c) Processamento:
 - ▲ Calcular o total da compra;
 - ▲ Atualiza o total de vendas assim como o status de 'F' para finalizada;

Considere os seguintes esquemas relacionais:

Compras(cod_compra, data_compra, tipo_pagamento)

Itens_Compras(cod_item, cod_compra, cod_produto, qtde, descrição, vlr_produto)

Vendas(cod_venda, nome_venda, total_venda, tipo_status)

Diagramação, inserir espaço de 10 linhas para a resposta.

Resposta Comentada

```

CREATE PROCEDURE SP_ProcessaVendas
(IN cod_compra_parm INT,
 cod_venda_parm INT)

```

```

BEGIN
    DECLARE total_venda_var DECIMAL(8,2);
    SET total_venda_var = (SELECT SUM(vlr_produtos * qtde) FROM
        ITEM_COMPRA
        WHERE cod_compra = cod_compra_parm);
    UPDATE VENDAS SET tipo_status = 'F',
        total_venda = total_venda_var
        WHERE cod_venda = cod_venda_parm;
END;

```

Fim da atividade

CURSOR

O termo é um acrônimo para Current Set Of Records (conjunto de registros concorrentes). São utilizados para posicionar um ponteiro em uma linha específica e podem permitir atualizações para as linhas com base na posição atual. Cursor é um recurso bastante interessante em bancos de dados pois permite que seus códigos SQL façam uma varredura de uma tabela ou consulta linha-por-linha, realizando mais de uma operação se for o caso.

Na maioria das vezes, um simples SELECT exibe na tela esta varredura, trazendo todos os registros da consulta em questão. A vantagem de usar um cursor é quando, além da exibição dos dados, queremos realizar algumas operações sobre os registros. Se o volume de operações for grande, fica muito mais fácil, limpo e prático escrever o código utilizando cursor, do que uma consulta SQL. Imagine sobre estoque de produtos, é muito mais vantajoso criar um cursor que faça a análise de cada produto de estoque, conferindo seu histórico, calculando sua previsão de vendas, capturando o melhor cliente do mês que já comprou, etc., do que utilizar o comando SELECT absurdamente imenso que talvez não consiga ainda todas as informações de forma simples.

Operações sobre Cursos:

- Declaração:** atribui um nome a um conjunto de valores recuperados a partir de um SELECT.

Sintaxe: DECLARE <nome do cursor> CURSOR FOR <comando SQL – SELECT>;

2. **Abertura:** abre o cursor para leitura, colocando o cursor na 1ºlinha do conjunto de dados recuperados.

Sintaxe: OPEN <nome do cursor>;

3. **Leitura de uma linha:** realiza a leitura da linha corrente e avança o cursor para a próxima linha;

Sintaxe: FETCH <nome do cursor> INTO <lista de variáveis>;

4. **Fechamento:** encerra o cursor.

Sintaxe: CLOSE <nome do cursor>;

5. **Identificando o fim do cursor:**

- ▲ A variável SQLSTATE, padrão do MySQL, assume valor '02000';
- ▲ Pode ser usado um handler(testa certas condições, como de erro).

Quando ocorrem erros em nossos procedimentos, em seguida, eles terminam sem completar a execução dos demais comandos. Para lidar com essas condições, temos que criar um manipulador (HANDLER FOR). A sintaxe geral do manipulador é a seguinte:

```
DECLARE tipo_handler HANDLER FOR condition_value[...] instrução
```

- ▲ Em primeiro lugar temos que usar **DECLARE** para criar um manipulador;
- ▲ tipo_handle: podem ser **continue**, **exit** ou **undo**.
 - ▲ **Continue:** a execução das rotinas atuais continuam depois da instrução handler.
 - ▲ **Exit:** a execução das rotinas atuais é terminada imediatamente.
 - ▲ **Undo:** é utilizado em tabelas transacionais para trabalhar reversão realizado até aquele momento.
- ▲ **Condition_value:**
 - ▲ **SQLWARNING:** é usado para mensagens de truncamento ou algum tipo de execução.
 - ▲ **SQLSTATE:** aparece quando um erro for enviado a um usuário e quando este executa uma operação ilegal ou que viole a integridade de dados.
- ▲ **Instrução:** instrução que será executada quando o handler é accionada.

Exemplo:

(1) Crie um stored procedure que utilize cursor chamado DEMO e realize todas as operações sobre cursores considerando o banco de dados TESTE das tabelas T1, T2 e T3.

```
CREATE PROCEDURE DEMO()
BEGIN
    DECLARE status INT DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET status = 1;

    DECLARE cur1 CURSOR FOR SELECT id, data FROM TESTE.T1;
    DECLARE cur2 CURSOR FOR SELECT i FROM TESTE.T2;
    DECLARE A CHAR(16);
    DECLARE B, C INT;
    OPEN cur1; OPEN cur2;
    REPEAT
        FETCH cur1 INTO A, B; FETCH cur2 INTO C;
        IF NOT status THEN
            IF B < C THEN
                INSERT INTO TESTE.T3 VALUES (A, B);
            ELSE
                INSERT INTO TESTE.T3 VALUES (A, C);
            END IF;
        END IF;
    UNTIL status END REPEAT;
    CLOSE cur1; CLOSE cur2;
END
```

onde:

- ▲ **DECLARE status INT DEFAULT 0:** declara e atribui o nome status tipo inteiro como valor 0.
- ▲ **DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET **status** = 1:** declara a variável de controle de looping do cursor. E SQLSTATE '02000' é uma condição não encontrada e isso ocorre quando REPEAT não pode continuar

porque não há mais linhas para percorrer e neste caso ao **status** é atribuído o valor de 1.

- ▲ `DECLARE cur1 CURSOR FOR SELECT id, data FROM TESTE.T1;`: declara nome do cursor cur1 ao conjunto de valores recuperados (id e data) do banco de dados TESTE da tabela T1.
- ▲ `DECLARE cur2 CURSOR FOR SELECT i FROM TESTE.T2;`: declara nome do cursor cur2 ao conjunto de valores recuperados (i do banco de dados TESTE da tabela T2).
- ▲ `DECLARE A CHAR(16);`: declara uma variável “A” do tipo caractere de tamanho 16.
- ▲ `DECLARE B, C INT;`: declara duas variáveis B e C do tipo inteiro.
- ▲ `OPEN cur1; OPEN cur2;`: abertura do cursor cur 1 e cur 2 para leitura, colocando o cursor na 1ºlinha do conjunto de dados recuperados.
- ▲ `REPEAT FETCHUNTIL status END REPEAT;`: observamos que o `FETCH` está dentro de um `REPEAT` por isso é repetido várias vezes até que o **status** seja verdadeiro como foi especificado em `UNTIL status END REPEAT`. Lembrar que **status** é definido inicialmente como falso (`DECLARE status INT DEFAULT 0`).
- ▲ `FETCH cur1 INTO A, B; FETCH cur2 INTO C;`: realiza o acesso ou leitura da linha corrente e avança os cursores para a próxima linha ou looping de execução do cur1 e cur2.
- ▲ `IF NOT status THEN`
 `IF B < C THEN`
 `INSERT INTO TESTE.T3 VALUES (A, B);`
 `ELSE`
 `INSERT INTO TESTE.T3 VALUES (A, C);`
 `END IF;`
`END IF;`
- ▲ `CLOSE cur1; CLOSE cur2;`: encerra os cursores cur 1 e cur 2.

Encontramos dois testes de avaliação.
No primeiro trata-se o controle de existir mais registros na tabela.
E no segundo temos o teste `B < C` dependendo se é verdadeiro ou

Início da atividade

Atividade 2 - Atende ao objetivo 2

Criar um stored procedure que contabilize todos os salários da tabela de EMPREGADOS utilizando o conceitos de cursor junto com as operações de

declaração, abertura, leitura linha por linha, fechamento e utilize o identificador como fim do cursor a variável SQLSTATE.

Diagramação, inserir espaço de 10 linhas para a resposta.

Resposta Comentada

```
CREATE PROCEDURE SP_sum_salaries
(OUT sum INT)
BEGIN
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE var_sum INT;
    DECLARE var_sal INT;
    DECLARE cur01 CURSOR FOR SELECT salario FROM EMPREGADOS;
    SET var_sum = 0;
    OPEN c;
    FETCH FROM cur01 INTO var_sal;
    WHILE (SQLSTATE = '00000') DO
        SET var_sum = var_sum + var_sal;
        FETCH FROM cur01 INTO var_sal;
    END WHILE;
    CLOSE cur01;
    SET sum = var_sum;
END;
```

Fim da atividade

TRIGGERS

O que é?

A Trigger (ou gatilho ou disparadores) é um tipo especial de procedimento que é disparado automaticamente quando ocorre uma mudança nos dados da tabelas. Eles diferem da Stored Procedures pelo fato de não serem chamados diretamente pelo usuário.

Pontos Importantes

- ▲ São vinculados a uma tabela, não existem “soltos” ou “separados” como procedimentos armazenados;
- ▲ São executados automaticamente na tentativa de alterar uma tabela (INSERT,

DELETE, UPDATE). Ele não pode ser executado manualmente.

Entre as vantagens da utilização da Trigger, podemos destacar:

- ▲ Criar validações que envolvam pesquisas em mais de uma tabela;
- ▲ Manter a integridade dos dados;
- ▲ Atualiza tabelas associadas;

Sintaxe:

```
CREATE TRIGGER <nome_trigger> <tempo_triggers> <evento_triggers> ON  
<nome_tabela> FOR EACH ROW  
BEGIN  
    <Comandos>  
END;
```

onde:

- ▲ Nome_trigger: define o nome do procedimento.
- ▲ tempo_triggers: define se o trigger será ativado de acordo com os eventos: AFTER e BEFORE.
 - BEFORE disparam antes das modificações da instrução serem aplicadas, e antes de qualquer restrição ser aplicada.
 - AFTER disparam após todas as restrições terem sido satisfeitas, e após todas as alterações terem sido aplicadas à tarefa de destino.
- ▲ evento_triggers: aqui se define qual será o evento: INSERT, UPDATE ou DELETE;
- ▲ nome_tabela: nome da tabela onde o triggers ficará “pendurado” aguardando o evento_triggers.
- ▲ Comandos – as instruções que o trigger deverá fazer quando for disparado;

Operadores OLD e NEW:

- ▲ Referem-se às tuplas responsáveis pelos eventos que dispararam o gatilho.
- ▲ Através desses operadores, é possível acessar os valores das tuplas envolvidas.
- ▲ O comportamento de OLD e NEW depende do evento disparado.
- ▲ INSERT: o operador NEW.nome_coluna permite verificar o valor enviado para

ser inserido em uma coluna de uma tabela. OLD.nome_coluna não está disponível.

- ▲ DELETE: o operador OLD.nome_coluna permite verificar o valor excluído ou a ser excluído. NEW.nome_coluna não está disponível.
- ▲ UPDATE: tanto OLD.nome_coluna quanto NEW.nome_coluna estão disponíveis, antes (BEFORE) ou depois (AFTER) da atualização de uma linha.

Visualizar as triggers: SHOW Nome_triggers;

Excluindo Triggers: DROP TRIGGERS Nome_trigger;

Exemplos:

(1) Será criada uma tabela de Usuários e uma tabela de Pontos. Cada usuário deve ter associado a si um registro dos seus pontos obtidos. Assim, cada inserção na tabela usuários implica em uma inserção na tabela de pontos. Considerando as seguintes tabelas:

| | |
|---|--|
| CREATE TABLE Usuários (id INT NOT NULL AUTO_INCREMENT, nome VARCHAR(50) NOT NULL, e-mail VARCHAR(30) NOT NULL, PRIMARY KEY (id)); | CREATE TABLE Pontos (id INT NOT NULL AUTO_INCREMENT, user_id INT NOT NULL, Pontos INT NOT NULL DEFAULT '0' PRIMARY KEY (id), FOREIGN KEY (user_id) REFERENCES Usuários(id)); |
|---|--|

Crie a trigger Log_atualiza após atualizações na tabela Pontos e realize a criação automática do registro na tabela de Pontos.

```
CREATE TRIGGER Log_atualiza AFTER INSERT ON Usuários  
FOR EACH ROW  
BEGIN  
    INSERT INTO Pontos SET user_id = NEW.id, pontos='0';  
END;
```

- (2) Crie um trigger tr_atualiza, que apaga a tabela2 cada vez que se altere a tabela Contas.

```
CREATE TRIGGER tr_atualiza AFTER INSERT ON Contas  
FOR EACH ROW  
BEGIN  
DELETE FROM tabela2;  
END
```

- (3) Crie um trigger chamado tr_item, que atualiza o campo valor da tabela VENDAS cada vez que se altere a tabela de ITENS.

```
CREATE TRIGGER tr_item AFTER INSERT ON ITENS  
FOR EACH ROW  
BEGIN  
UPDATE VENDAS set valor = valor + New.ValorTotal  
WHERE Pedido = New.Pedido;  
END
```

- (4) Realize a exclusão da tabela ITENS, após da cada atualização.

```
CREATE TRIGGER trDel_Item AFTER DELETE on ITENS  
BEGIN  
UPDATE VENDAS set valor = valor - OLD.ValorTotal  
WHERE Pedido = OLD.Pedido;  
END
```

- (5) Atualização da quilometragem rodada por um carro **após** sua devolução na tabela de CARROS.

```
CREATE TRIGGER tr_kmrodados after UPDATE on RESERVA  
FOR EACH ROW  
BEGIN
```

```
UPDATE CARROS SET quilometragem = quilometragem +NEW.quilometragem  
WHERE NEW.carro_reserva = id_carro;  
' END;
```

Início da atividade

Atividade 3 - Atende ao objetivo 3

Imagine que você está num projeto que possui um conjunto de atividades. Cada atividade possui quatro tempos a seguir: DataCedoComecar, DataCedoTerminar, DataTardeComecar e DataTardeTerminar. Precisamos planejar as datas mais tarde para começar e terminar das atividades e guardar num histórico as datas anteriores e o motivo desse planejamento.

Diagramação, inserir espaço de 10 linhas para a resposta.

Resposta Comentada

```
CREATE TRIGGER Log_replanejamento AFTER UPDATE ON ATIVIDADES
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF OLD.dt_tarde_inicio <> NEW.dt_tarde_inicio OR  
OLD.dt_tarde_fim <> NEW.dt_tarde_fim
```

```
THEN
```

```
INSERT INTO Replanejamentos SET atividade_id = OLD.id,  
dt_cedo_inicio = OLD.dt_cedo_inicio,  
dt_cedo_fim = OLD.dt_cedo_fim,  
dt_tarde_inicio = OLD.dt_tarde_inicio,  
dt_tarde_fim = OLD.dt_tarde_fim,  
dt_replanejamento = NOW(),  
observacao = OLD.observacao;
```

```
END IF;
```

```
END;
```

Foi criado a trigger Log_replanejamento. Para cada linha, se a data mais tarde para começar for diferente da nova data OU se a data mais tarde para terminar for diferente, insira na tabela Replanejamento.

Fim da atividade

Inicio Atividade Online

Acesse o ambiente virtual e resolva a atividade a seguir em MySQL, enviando sua resposta ao tutor. Escreva uma procedure que receba como parâmetro o número do pedido e um número que represente a quantidade de parcelas em que este pedido será dividido. A procedure deve obter o valor total deste pedido, calcular o valor de cada parcela e gravar cada parcela na tabela Parcelas.

Se a quantidade de parcelas ultrapassar 3, acrescente 10% ao valor total do pedido, divida-o na quantidade de parcelas recebida como parâmetro e grave-as na tabela Parcelas. Se a quantidade de parcelas for 1, retorne a mensagem: pedido à vista e interrompa o processamento. Não deixe que o número de parcelas ultrapasse o 10. Se ultrapassar, retorne a mensagem: Quantidade de parcelas inválida.

Fim da Atividade Online

Conclusão

SQL Trigger é uma instrução SQL ou um conjunto de instruções SQL, que fica armazenado, é será ativado ou disparado quando um evento ocorrer numa tabela. O evento pode ser INSERT, UPDATE ou DELETE. Às vezes, um trigger é referido como um tipo especial de stored procedure em termos de código dentro de seu corpo. A diferença entre um trigger e um stored procedure é que o trigger é ativado ou chamado quando um evento acontece numa tabela, um stored procedure deve ser chamado explicitamente. Por exemplo, podemos fazer algumas ações na base de dados antes ou depois de inserir um novo registo.

Próxima Aula

Todas as informações estratégicas de uma empresa, de uma instituição ou de um governo dependem da segurança do banco de dados onde estão armazenadas. Se esta segurança é de algum modo quebrada, seja acidental ou propositadamente, os resultados podem ser extremamente danosos. Por este motivo, a segurança dos bancos de dados é uma questão importante tanto para os projetistas dos bancos de

dados quanto para os usuários. Na próxima aula estudaremos os controles de acesso aos dados.

Resumo

Stored Procedure.

O que é?

Stored Procedure ou procedimentos armazenados são uma coleção de comando em SQL, compilados e armazenados no servidor. Geralmente eles representam tarefas repetitivas e possuem parâmetros de entrada ou saída. A Stored Procedure serve para:

- ▲ melhorar o tráfego na rede,
- ▲ melhorar a performance das aplicações,
- ▲ criar mecanismos de segurança e
- ▲ melhorar a manutenção dos códigos SQL e das aplicações que acessam o banco de dados.

Ao executar a procedure pela 1ºvez ela é compilada e a cada execução seus resultados são colocados em cache; o cache é semelhante a uma memória que guarda as últimas operações do banco de dados, caso a procedure seja executada novamente ela pega o resultado do cache diminuindo o esforço do banco de dados e aumentando a velocidade de acesso.

Quando utilizar procedures?

- ▲ Quando temos várias aplicações escritas em diferentes linguagens ou rodam em plataformas diferentes, porém todas elas executam a mesma função.
- ▲ Quando damos prioridade à consistência e segurança.

Exemplo: Os bancos (Itaú, Bradesco, Real, etc), em geral, utilizam Stored Procedures para todas as operações em comum. Os procedimentos podem assegurar que as operações sejam registradas de forma correta e segura.

Por que é mais seguro?

Seguindo a linha de raciocínio dos bancos, utilizando Stored Procedures outras aplicações e usuários não conseguiriam nenhum tipo de acesso às tabelas do banco de dados de forma direta. Eles poderiam apenas executar as Stored Procedures, que rodam ações específicas e determinadas pelos DBAs e desenvolvedores.

Como criar a Stored Procedure?

Antes de criar esta rotina, é necessário mudar o delimitador de comandos do MySQL, pois no conteúdo da rotina existe o caractere “ ; ”, que não pode ser confundido pelo sistema como uma finalização de comando. Não esqueça de executar o delimitador: Delimiter //.

Vejamos a sintaxe de criação da stored procedure:

CREATE PROCEDURE <banco de dados>.<nome da rotina>

(Parâmetros de entrada e saída) <nome da variável> <tipo de dados>
BEGIN
<comandos Select into>
END:

onde:

- ▲ CREATE PROCEDURE <banco de dados>.<nome da rotina>: cria a stored procedure de nome da rotina no banco de dados que já existe.
- ▲ (Parâmetros de entrada e saída) <nome da variável> <tipo de dados>: o parâmetro de entrada IN, o parâmetro de saída OUT e INOUT as duas coisas. Nome da variável criada e o tipo de dados que corresponde a variável criada.
- ▲ BEGIN ... END: estrutura da stored procedure que contém os comandos a serem executados. Aqui no meio vai as instruções SQL.
- ▲ Pode-se utilizar os comandos DECLARE (declaração de variáveis locais) e SET (atribuição de valores). Permite a execução de instruções if-then-else, loops, etc; e comandos de select, insert, delete e update podem ser utilizados dentro de uma procedure

Podemos voltar o delimitador para o padrão. Agora que a rotina já está criada e incluída no banco de dados, para executá-la e armazenar os valores em variáveis, basta utilizar o comando CALL.

Como alterar uma Stored Procedure?

ALTER PROCEDURE, este comando pode ser usado para renomear uma stored procedure e para alterar suas características. Basta acessar a sua base de dados e criar uma nova query e digitar os comandos responsáveis pela alteração e executá-los.

Sintaxe:

ALTER PROCEDURE <nome da rotina> [características ...]

características:

```
NAME novo nome  
| SQL SECURITY {DEFINER | INVOKER}  
| COMMENT 'string'
```

onde:

- ▲ A característica SQL SECURITY:
 - ▲ SECURITY INVOKER indica que a stored procedure deve ser executada com os privilégios do usuário a chamou. Este é o padrão.
 - ▲ SECURITY DEFINER especifica que a stored procedure deve ser executada com os privilégios do usuário que a criou.
- ▲ A cláusula COMMENT é uma extensão do MySQL, e pode ser usada para descrever a stored procedure. Esta informação é exibida pelas instruções SHOW CREATE PROCEDURE.

Ex: ALTER PROCEDURE SPEmpregado SQL SECURITY DEFINER;

Acabamos de alterar a stored procedure SPEmpregado para que seja executada com os privilégios do usuário que a criou.

Como excluir uma Stored Procedure?

DROP PROCEDURE é usado para deletar uma stored procedure. Isto é, a procedure especificada é removida do servidor.

```
DROP PROCEDURE [IF EXISTS] <nome da rotina>
```

A cláusula IF EXISTS é uma extensão do MySQL. Ela previne que um erro ocorra se o procedimento não existe. Um aviso é produzido e pode ser visualizado com SHOW WARNINGS. Ex: DROP PROCEDURE IF EXISTS SPEmpregado;

CURSOR

O termo é um acrônimo para Current Set Of Records (conjunto de registros concorrentes). São utilizados para posicionar um ponteiro em uma linha específica e podem permitir atualizações para as linhas com base na posição atual. Cursor é um

recurso bastante interessante em bancos de dados pois permite que seus códigos SQL façam uma varredura de uma tabela ou consulta linha-por-linha, realizando mais de uma operação se for o caso.

Na maioria das vezes, um simples SELECT exibe na tela esta varredura, trazendo todos os registros da consulta em questão. A vantagem de usar um cursor é quando, além da exibição dos dados, queremos realizar algumas operações sobre os registros. Se o volume de operações for grande, fica muito mais fácil, limpo e prático escrever o código utilizando cursor, do que uma consulta SQL. Imagine sobre estoque de produtos, é muito mais vantajoso criar um cursor que faça a análise de cada produto de estoque, conferindo seu histórico, calculando sua previsão de vendas, capturando o melhor cliente do mês que já comprou, etc., do que utilizar o comando SELECT absurdamente imenso que talvez não consiga ainda todas as informações de forma simples.

Operações sobre Cursos:

1. **Declaração:** atribui um nome um conjunto de valores recuperados a partir de um SELECT.

Sintaxe: DECLARE <nome do cursor> CURSOR FOR <comando SQL – SELECT>;

2. **Abertura:** abre o cursor para leitura, colocando o cursor na 1^alinha do conjunto de dados recuperados.

Sintaxe: OPEN <nome do cursor>;

3. **Leitura de uma linha:** realiza a leitura da linha corrente e avança o cursor para a próxima linha;

Sintaxe: FETCH <nome do cursor> INTO <lista de variáveis>;

4. **Fechamento:** encerra o cursor.

Sintaxe: CLOSE <nome do cursor>;

5. **Identificando o fim do cursor:**

- ▲ A variável SQLSTATE, padrão do MySQL, assume valor '02000';
- ▲ Pode ser usado um handler(testa certas condições, como de erro).

Quando ocorrem erros em nossos procedimentos, em seguida, eles terminam sem completar a execução dos demais comandos. Para lidar com essas condições, temos que criar um manipulador (HANDLER FOR). A sintaxe geral

do manipulador é a seguinte:

```
DECLARE tipo_handler HANDLER FOR condition_value[,...] instrução
```

- ▲ Em primeiro lugar temos que usar **DECLARE** para criar um manipulador;
- ▲ tipo_handle: podem ser **continue**, **exit** ou **undo**.
 - ▲ **Continue**: a execução das rotinas atuais continuam depois da instrução handler.
 - ▲ **Exit**: a execução das rotinas atuais é terminada imediatamente.
 - ▲ **Undo**: é utilizado em tabelas transacionais para trabalhar reversão realizado até aquele momento.
- ▲ **Condition_value**:
 - ▲ **SQLWARNING**: é usado para mensagens de truncamento ou algum tipo de execução.
 - ▲ **SQLSTATE**: aparece quando um erro for enviado a um usuário e quando este executa uma operação ilegal ou que viole a integridade de dados.
- ▲ **Instrução**: instrução que será executada quando o handler é accionada.

TRIGGERS

O que é?

A Trigger (ou gatilho ou disparadores) é um tipo especial de procedimento que é disparado automaticamente quando ocorre uma mudança nos dados da tabelas. Eles diferem da Stored Procedures pelo fato de não serem chamados diretamente pelo usuário.

Pontos Importantes

- ▲ São vinculados a uma tabela, não existem “soltos” ou “separados” como procedimentos armazenados;
- ▲ São executados automaticamente na tentativa de alterar uma tabela (INSERT, DELETE, UPDATE). Ele não pode ser executado manualmente.

Entre as vantagens da utilização da Trigger, podemos destacar:

- ▲ Criar validações que envolvam pesquisas em mais de uma tabela;

- ▲ Manter a integridade dos dados;
- ▲ Atualiza tabelas associadas;

Sintaxe:

```
CREATE TRIGGER <nome_trigger> <tempo_triggers> <evento_triggers> ON
<nome_tabela> FOR EACH ROW
BEGIN
    <Comandos>
END;
```

onde:

- ▲ Nome_trigger: define o nome do procedimento.
- ▲ tempo_triggers: define se o trigger será ativado de acordo com os eventos: AFTER e BEFORE.
 - BEFORE disparam antes das modificações da instrução serem aplicadas, e antes de qualquer restrição ser aplicada.
 - AFTER disparam após todas as restrições terem sido satisfeitas, e após todas as alterações terem sido aplicadas à tarefa de destino.
- ▲ evento_triggers: aqui se define qual será o evento: INSERT, UPDATE ou DELETE;
- ▲ nome_tabela: nome da tabela onde o triggers ficará “pendurado” aguardando o evento_triggers.
- ▲ Comandos – as instruções que o trigger deverá fazer quando for disparado;

Operadores OLD e NEW:

- ▲ Referem-se às tuplas responsáveis pelos eventos que dispararam o gatilho.
- ▲ Através desses operadores, é possível acessar os valores das tuplas envolvidas.
- ▲ O comportamento de OLD e NEW depende do evento disparado.
- ▲ INSERT: o operador NEW.nome_coluna permite verificar o valor enviado para ser inserido em uma coluna de uma tabela. OLD.nome_coluna não está disponível.
- ▲ DELETE: o operador OLD.nome_coluna permite verificar o valor excluído ou a ser excluído. NEW.nome_coluna não está disponível.
- ▲ UPDATE: tanto OLD.nome_coluna quanto NEW.nome_coluna estão disponíveis,

antes (BEFORE) ou depois (AFTER) da atualização de uma linha.

Visualizar as triggers: SHOW Nome_triggers;

Excluindo Triggers: DROP TRIGGERS Nome_trigger;

Referências Bibliográficas

CARVALHO C.R., 2006. SQL-Guia Prático. 2ed. Rio de Janeiro. Brasport.

DATE C.J., 2003. Introdução a Sistemas de Bancos de Dados. 8ed. Americana. Rio de Janeiro. Elsevier.

ELMASRI R., Navathe S., 2009. Sistemas de Banco de Dados. 5ed. São Paulo. Pearson Addison Wesley.

HEUSER C.A. 2009. Projeto de Banco de Dados. 4ed. Porto Alegre. Bookman.

SETZER V.W. & Corrêa da Silva F.S. 2005. Bancos de Dados. 1ed. São Paulo. Edgard Blucher.

SILBERSCHATZ A., Korth H., 2008. Sistema de Banco de Dados. 3ed. São Paulo. Pearson

Modelando e Implementando Bancos de Dados Relacionais

Professores: Cássia Blondet Baruque, Lúcia Blondet Baruque, Rubens Nascimento Melo

Aula 14

Direitos de acesso

Meta

Apresentar os comandos importantes para atribuir os direitos de acesso ao sistema e como conceder grupos de direitos de acesso.

Objetivos

Esperamos que, ao final desta aula, você seja capaz de:

1. Aplicar os direitos de acesso ao sistema:

- ▲ Atribuir privilégios de sistema: GRANT
- ▲ Revogar privilégios de sistema: REVOKE

2. Conceder grupos de direitos de acesso:

- ▲ Criar *role*: CREATE ROLE
- ▲ Remover *role*: DROP ROLE

Pré-requisitos

Para ter um bom aproveitamento desta aula, é importante você relembrar os comandos relacionados à linguagem SQL aprendidos nas Aulas 8, 9 e 10.

Introdução

Protegendo seus dados...

Atualmente, as pessoas prezam muito a segurança, que é considerada prioridade.

Melhorar a segurança de casa, carro, conta bancária é uma preocupação diária na nossa sociedade. Proteger seu patrimônio investindo em planos de seguro se faz muito importante. Dessa forma, você poderá sair com o seu carro 0 km ou usado em



qualquer lugar, com total segurança, tranquilidade e agilidade! Então, por que não se preocupar também em proteger os dados do seu banco de dados? É necessário impedir o acesso das informações contidas no banco de dados aos usuários.

O controle de privilégios é um importante mecanismo existente em SGBD. Com ele é possível garantir que os usuários realizem apenas as operações que lhes são permitidas. Todo acesso a um SGBD é realizado com a identificação de um usuário, de forma implícita ou explícita. A cada usuário podem estar associados diversos privilégios e, de acordo com os privilégios que possui, o usuário pode, por exemplo:

- conectar-se ao banco de dados;
- criar outros usuários;
- ler, alterar, incluir ou apagar dados de uma ou mais tabelas;
- configurar o sistema de forma que um determinado usuário não saiba a existência de uma ou mais tabelas, etc.

<http://www.sxc.hu/photo/352344> Autor: Marc and Cristina Palmer and Burke

Por outro lado, de forma a facilitar o gerenciamento e a atribuição de privilégios a usuários, o padrão SQL define a possibilidade de conceder grupos de direitos de acesso (*roles*), os quais podem ser tratados como conjuntos de privilégios que podem ser atribuídos aos diferentes usuários.



Figura 14.1: Controle do acesso do usuário

Ilustração, favor melhorar a imagem acima.

A fim de melhorar os mecanismos de controle de acesso e segurança, os gerenciadores de banco de dados apresentam diversos privilégios (que serão vistos nesta aula), além dos especificados no padrão (como senha, por exemplo). Cada SGBD possui seu próprio modelo de privilégios. Nesses modelos, os privilégios são classificados geralmente como privilégios de objeto e privilégios de sistema. Dessa forma, são comuns em SGBD privilégios específicos como:

- conectar-se a uma instância do SGBD ou a uma de suas bases de dados;
- criar e destruir objetos do banco de dados como tabelas, índices, visões, gatilhos e procedimentos armazenados, entre outros;
- consultar, atualizar, incluir e excluir dados em tabelas (e em visões, para os gerenciadores que suportam a realização dessas operações nas mesmas);
- executar procedimentos e funções armazenados;
- alterar as características do sistema e de suas bases de dados, tais como parâmetros de inicialização;
- realizar operações de geração de cópias de segurança (*backup*) e recuperação de tais cópias (*restore*).

Em geral, o usuário, o analista de sistemas ou o programador não necessitarão de todos esses privilégios. A maioria desses privilégios se relaciona com ações realizadas especificamente pelo administrador do banco de dados (DBA). Usualmente, é o DBA que concede os privilégios necessários para que os outros usuários realizem suas ações.

Nesta aula você aprenderá como os bancos de dados e os objetos inseridos nele podem ter mais segurança e como se pode ter um completo controle sobre quem pode fazer o que com seus dados.



Figura 14.2: É preciso proteger os dados e restringir os privilégios de quem acessa o banco de dados

<http://www.sxc.hu/photo/913643> Autor: Vangelis Thomaidis

Fim da introdução

Direito de sistema

Trata-se de como obter acesso ao banco de dados. Mas antes disso precisamos aprender como criar e remover usuários. Quando o usuário for criado, o DBA poderá conceder privilégios de sistema específicos para ele.

O padrão SQL define o conceito de identificadores de usuários como a representação no SGBD de usuários do mundo real. Dessa forma, identificadores de usuários representam usuários do SGBD. Tanto a criação quanto a remoção de usuários são operações bastante dependentes da implementação do SGBD.

O DBA cria os usuários empregando a instrução CREATE USER da seguinte maneira:

```
CREATE USER usuário IDENTIFIED BY senha;
```

Exemplo: Luis Santos, analista de sistemas, acabou de chegar à empresa e, para poder ingressar no sistema, o DBA deve criar sua senha e identificação como usuário.

Para isso, ele realiza os seguintes comandos no MySQL:

```
CREATE USER Luis IDENTIFIED BY lysts20;
```

Observação:

- ▲ Por padrão, o primeiro usuário – o usuário *root* (raiz) – tem controle completo sobre qualquer coisa no banco de dados. Isso é importante porque o usuário raiz precisa estar apto a criar contas de usuários para todos os outros usuários. Em MySQL, a criação do usuário *root* é feito da seguinte forma:

```
SET PASSWORD FOR root@localhost = PASSWORD('a5hjp12qzy');
```

onde:

- root: o nome de usuário raiz é simplesmente root;
- localhost: indica o lugar onde o sistema SQL está instalado e sendo executado. Se você estiver utilizando uma máquina cliente em algum outro lugar, você poderá usar root@winequals.light.com, mas este é um exemplo de servidor.

Atribuir privilégios de sistema

O DBA permite ao usuário controlar todos os objetos do banco de dados. É um direito que deve ser concedido com muito crédito para evitar problemas ao banco de dados. Recomenda-se que apenas o administrador do banco de dados o possua.

Então, quando o usuário for criado, o DBA poderá conceder privilégios de sistema específicos para ele. A sintaxe dos direitos de acesso no MySQL é:

```
GRANT <tipo de privilégios> ON {NomeBanco.tabela | NomeBanco.* | *.*}

TO <nome usuario1> @ <dominio> [IDENTIFY BY <senha>] [, nome
usuario2 [IDENTIFY BY <senha2>] ...

[WITH GRANT OPTION]
```

No comando acima, as chaves [] indicam que o comando é opcional. O primeiro item a ser informado é(são) o(s) privilégio(s) a ser(em) concedido(s) ao(s) usuário(s). A lista de privilégios do sistema existentes no MySQL é apresentada na tabela a seguir:

| Privilégio | Descrição |
|------------------|--|
| ALL [PRIVILEGES] | Todos os privilégios exceto GRANT OPTION |
| ALTER | Permite executar ALTER TABLE. Permite alterar a estrutura de tabelas. |
| CREATE | Permite executar CREATE TABLE. Permite criar banco de dados e tabelas. |

| | |
|-------------------------|---|
| CREATE TEMPORARY TABLES | Permite executar CREATE TEMPORARY TABLE. Permite a criação de tabelas temporárias em expressões SQL que utilizam esse recurso. |
| DELETE | Permite executar DELETE. Permite excluir informações. |
| DROP | Permite executar DROP TABLE. Permite excluir estruturas (bases e tabelas). |
| EXECUTE | Permite executar <i>stored procedures</i> (MySQL 5.0). |
| INDEX | Permite o gerenciamento de índices. Permite executar CREATE INDEX e DROP INDEX. |
| INSERT | Permite inserir informações em tabelas. |
| LOCK TABLES | Permite bloquear tabelas. |
| PROCESS | Permite visualizar e finalizar processos do MySQL. |
| RELOAD | Permite recarregar bancos de dados. Permite executar FLUSH. |
| REPLICATION CLIENT | Permite solicitar replicação. Permite ao usuário obter a localização do Master ou Slave. |
| REPLICATION SLAVE | Permite replicar suas informações. Necessário para a replicação Slave (leitura dos eventos do log binário do Master) |
| SELECT | Permite consultas. |
| SHOW DATABASES | Permite visualizar todas as estruturas dos bancos existentes. |
| SHUTDOWN | Permite desligar o servidor MySQL. |
| SUPER | Permite configurar os dados do servidor MASTER (em caso de replicação). Permite executar CHANGE MASTER, KILL, PURGE MASTER LOGS e SET GLOBAL. Permite conectar-se ao servidor uma vez, mesmo que o max_connections tenha sido atingido. |
| UPDATE | Permite alterar informações em tabelas. Permite executar UPDATE. |
| GRANT OPTION | Permite ao usuário repassar os seus privilégios. |
| INSERT | Permite executar INSERT em tabelas. |

Tabela 14.1: Lista de privilégios no MySQL

Uma vez informados os privilégios do usuário, você deverá indicar o nível ao qual o privilégio se aplica, sendo possível especificar três níveis. Veja a tabela a seguir:

| | | |
|----------|--------------|---|
| 1 | *.* | Privilégio global |
| 2 | NomeBanco.* | Qualquer tabela do banco com nome “NomeBanco”. |
| 3 | NomeBanco.tb | Apenas a tabela tb do banco de dados “NomeBanco”. |

Depois do nível, você deverá indicar o usuário ou a lista de usuários para os quais os privilégios se aplicam.

Exemplos:

- (1) Consideremos permitir à usuária MARIA realizar quaisquer operações na tabela chamada EDITORA. Para atribuir à referida usuária tais privilégios, podemos utilizar a cláusula ALL PRIVILEGES (ALL PRIVILEGES permite ao usuário todos os privilégios, exceto o *grant option*, que é outro privilégio), como:

GRANT ALL PRIVILEGIES ON EDITORA TO MARIA;

- (2) Agora atribuiremos à referida usuária os mesmos privilégios do administrador (root):

**GRANT ALL PRIVILEGES ON *.* TO MARIA@localhost
IDENTIFIED BY 'passwordmaria';**

- (3) Outra situação comum é no caso dos webmasters Luis e Márcio; eles, em cada site, vão precisar de uma ou mais bases de dados e, naturalmente, cada um vai precisar de um *login* próprio com acesso apenas às suas próprias bases de dados.

GRANT ALL ON BD1_php.* TO Luis IDENTIFIED BY 'passwordluis';

GRANT ALL ON BD2_php.* TO Márcio IDENTIFIED BY 'passwordmarcio';

isso:

- Permite tudo na base BD1_php para o usuário Luis, identificado pela senha passwordluis.
 - Permite tudo na base BD2_php para o usuário Márcio, identificado pela senha passwordmarcio.
- (4) Consideremos permitir que todos os usuários no sistema consultem dados na tabela DISCIPLINA.
- GRANT SELECT ON DISCIPLINA TO PUBLIC;**
- (5) Consideremos conceder ao usuário João a possibilidade de consultar qualquer tabela do banco de dados PRODUTO, ou seja, somente leitura:
- GRANT SELECT ON PRODUTO.* TO João@'%' IDENTIFIED BY vend123;**
- O parâmetro "%" representa qualquer domínio, ou seja, o usuário João pode consultar toda a tabela de "PRODUTO" mediante a senha "vend123".
- (6) Caso você deseje consultar os privilégios de um usuário, pode utilizar o seguinte comando:
- SHOW GRANTS FOR 'estagiario'@'%';**

(7) Sempre que mudar/"setar" algum tipo de permissão no MySQL, faz-se necessário utilizar o comando que atualiza a lista de privilégios: **FLUSH PRIVILEGES;**

Observações:

- ▲ Para que os comandos GRANT apresentados nos exemplos possam ser executados com sucesso, devem ser realizados por um usuário que possua os privilégios adequados;
- ▲ O AD (administrador de dados) possui privilégios de sistema que lhe permitem realizar todas ou quase todas as operações do banco de dados. Mas um usuário que tenha recebido tais privilégios com a opção WITH GRANT OPTION também pode realizar essas operações.
- ▲ O comando FLUSH PRIVILEGES faz com que o servidor MySQL atualize as tabelas de permissões, de modo que a alteração entre em vigor

automaticamente, sem precisar reiniciar o servidor. Na verdade, ele não é necessário ao adicionar usuários usando o comando GRANT, mas é bom se acostumar a utilizá-lo sempre que usar comandos que modifiquem as permissões de acesso. Você vai notar que a maioria dos tutoriais inclui o comando depois das operações relacionadas a alterações nas permissões de acesso;

- ▲ O padrão SQL define um privilégio como sendo uma autorização para que uma dada ação seja executada. Essas ações geralmente incidem sobre um objeto de banco de dados, como tabelas, visões, gatilhos ou colunas.

Início da atividade

Atividade 1 - Atende ao objetivo 1

O diretor de uma empresa, preocupado com a possibilidade de novos ou antigos funcionários do departamento de pessoal alterarem algo incorretamente, ou, ainda pior, deletar dados, deseja proteger a tabela DEVEDORES.

Considere o esquema relacional da tabela DEVEDORES como:

DEVEDORES (código, nome_representante, empresa, total_devedor, data, tipo_acordo, situação, observação).

Para proteger o banco de dados, ele solicita as seguintes atividades:

- 1) Criar senha e identificação de dois funcionários novos;
- 2) Os funcionários novos podem ter acesso ao sistema, mas é só isso. Agora, um deles precisará consultar as colunas de código, nome_representante e empresa e o outro funcionário deverá utilizar apenas a coluna empresa, ambos da tabela DEVEDORES. Crie as permissões correspondentes;
- 3) Os novos funcionários não devem inserir as informações total_devedor, data, tipo_acordo, situação e observação;
- 4) Não poderão atualizar as informações total_devedor, data, tipo_acordo, situação e observação;

- 5) Conceder privilégios de consulta na tabela EMPREGADOS para os funcionários Luis e Maria;
- 6) Conceder privilégios para atualizar as colunas dname e date na tabela DEPT_PESSOAL aos usuários Luis e Maria;
- 7) Conceder ao funcionário Luis os privilégios de pesquisa e inserir na tabela DEPT_PESSOAL. Logo, dar autoridade ao usuário Luis para passar os privilégios;
- 8) Permitir que todos os usuários do sistema consultem dados na tabela DEPT_PESSOAL;
- 9) Atribuir direitos de execução de *procedures* para usuários, mas somente se o banco de dados permitir tal operação.

Diagramação, inserir 10 linhas para a resposta

Resposta comentada

- 1) CREATE USER *nome_usuário1* IDENTIFIED BY *muser23*;
CREATE USER *nome_usuário2* IDENTIFIED BY *guser17*;
- 2) GRANT SELECT (código, nome_representante, empresa) ON DEVEDORES
TO *muser23*;
GRANT SELECT (empresa) ON DEVEDORES TO *guser17*;
- 3) GRANT INSERT (código, nome_representante, empresa) ON DEVEDORES
TO *muser23, guser17*;
- 4) GRANT UPDATE (código, nome_representante, empresa) ON DEVEDORES TO
muser23, guser17;
- 5) GRANT SELECT ON EMPREGADOS TO Luis, Maria;
- 6) GRANT UPDATE (dname, date) ON DEPT_PESSOAL TO Luis, Maria;
- 7) GRANT SELECT, INSERT ON DEPT_PESSOAL TO Luis WITH GRANT
OPTION;
- 8) GRANT SELECT ON DEPT_PESSOAL TO PUBLIC;
- 9) GRANT EXECUTE ON PROCEDURE TESTE;

Fim da atividade

Revogar privilégios de sistema

Após o usuário ter recebido o privilégio para executar uma ação, ele irá manter esse privilégio até que ele seja explicitamente revogado. Para revogar um ou mais privilégios de um usuário, deve-se utilizar o comando REVOKE da seguinte maneira:

```
REVOKE [GRANT OPTION FOR] <Privilégios> ON objeto FROM  
nome_usuário [CASCADE CONSTRAINTS];
```

Onde:

- ▲ REVOKE e FROM: Fazem parte do comando;
- ▲ GRANT OPTION FOR: É opcional. Indica que a habilidade de conceder o privilégio em questão para outros usuários será revogada;
- ▲ Privilégios: Lista de privilégios que está sendo retirada (quando a cláusula GRANT OPTION FOR não for especificada) ou lista de privilégios cuja habilidade de propagar a outros usuários está sendo retirada (quando a cláusula GRANT OPTION FOR utilizada);
- ▲ ON objeto: Refere-se ao nome do banco de dados ou tabela em questão;
- ▲ nome_usuário: Nome de um ou mais usuários ou papéis para quem o privilégio será revogado, separados por vírgula.

Exemplos:

(1) Consideremos que o usuário MARIA não deva mais possuir a possibilidade de alterar dados da tabela EDITORA. Para revogar esse privilégio, pode-se utilizar o comando a seguir:

```
REVOKE UPDATE ON EDITORA FROM MARIA;
```

(2) Imagine agora que não deve mais ser permitido ao usuário LUCAS atribuir a outros usuários o privilégio de atualização na tabela AUTOR, mas manter a possibilidade de que o referido usuário possa atualizar dados na tabela AUTOR:

```
REVOKE GRANT OPTION FOR UPDATE ON AUTOR FROM LUCAS;
```

(3) Considere a situação em que não deve ser permitido mais ao usuário Lucas realizar quaisquer ações na tabela ASSUNTO. Para tal, pode-se executar o seguinte comando:

```
REVOKE ALL PRIVILEGES ON ASSUNTO FROM LUCAS;
```

Assim, retiram-se do usuário os privilégios para executar ações na tabela ASSUNTO.

(4) Caso um estagiário esteja realizando muitas consultas, podemos retirar os direitos de acesso com o seguinte comando:

```
REVOKE SELECT ON minha_base.* FROM 'estagiario'@'%';
```

(5) Se for necessário, podemos ainda excluir ou alterar a senha desse usuário:

```
SET PASSWORD FOR 'estagiário' = PASSWORD('nova_senha');
```

```
DROP USER 'estagiário';
```

Início da atividade

Atividade 2 - Atende ao objetivo 1

Agora, escreva o comando REVOKE para os seguintes enunciados:

- 1) O usuário Felipe não deve mais possuir a possibilidade de alterar dados da tabela DEVEDORES.
- 2) Além disso, não deve mais ser permitido ao usuário Felipe atribuir a outros usuários o privilégio de atualização na tabela DEVEDORES.
- 3) Considerar a situação em que não é mais permitido ao usuário Marco realizar quaisquer ações na tabela DEVEDORES.
- 4) Revogar os direitos de execução de *procedures* para usuários se o banco de dados permitir tal operação:
- 5) Revogar os privilégios SELECT e INSERT fornecidos ao usuário Luis na tabela DEPT_PESSOAL.
- 6) Revogar os privilégios de consulta e inserir na tabela PESSOAS aos funcionários José e Pedro.
- 7) Revogar o privilégio de atualizar o campo salário da tabela FOLHAPAGTO do usuário Ana Karla.

Diagramação, inserir 7 linhas para a resposta

Resposta comentada

- 1) REVOKE UPDATE ON DEVEDORES FROM Felipe;
- 2) REVOKE GRANT OPTION FOR UPDATE ON DEVEDORES FROM Felipe;
- 3) REVOKE ALL PRIVILEGES ON DEVEDORES FROM Marco;
- 4) REVOKE EXECUTE ON PROCEDURE TESTE;
- 5) REVOKE SELECT, INSERT ON DEPT_PESSOAL FROM Luis;
- 6) REVOKE SELECT, INSERT ON PESSOAS FROM JOSE,PEDRO;
- 7) REVOKE UPDATE(SALARIO) ON FOLHAPAGTO FROM AnaKarla;

Fim da atividade

Conceder grupos de direitos de acesso

Num ambiente de banco de dados de produção podem existir diversos usuários e algumas centenas ou até milhares de tabelas, entre outros objetos. Muitos dos usuários podem utilizar o mesmo conjunto de objetos, devendo ter privilégios para executar conjuntos similares de ações.

Atribuir privilégios referentes a cada um dos objetos para cada usuário de forma individual pode ser uma tarefa bastante trabalhosa. Além disso, à medida que aumenta o número de tabelas ou outros objetos sobre os quais cada usuário deve ter privilégios, aumenta também a probabilidade de que, por erro ou esquecimento, um privilégio necessário não seja atribuído, podendo em alguns casos causar graves problemas. Para facilitar as tarefas de administração do banco de dados relacionadas com a atribuição e revogação de privilégios e reduzir o número de erros nessas operações, foi definido o conceito de *roles* (funções).

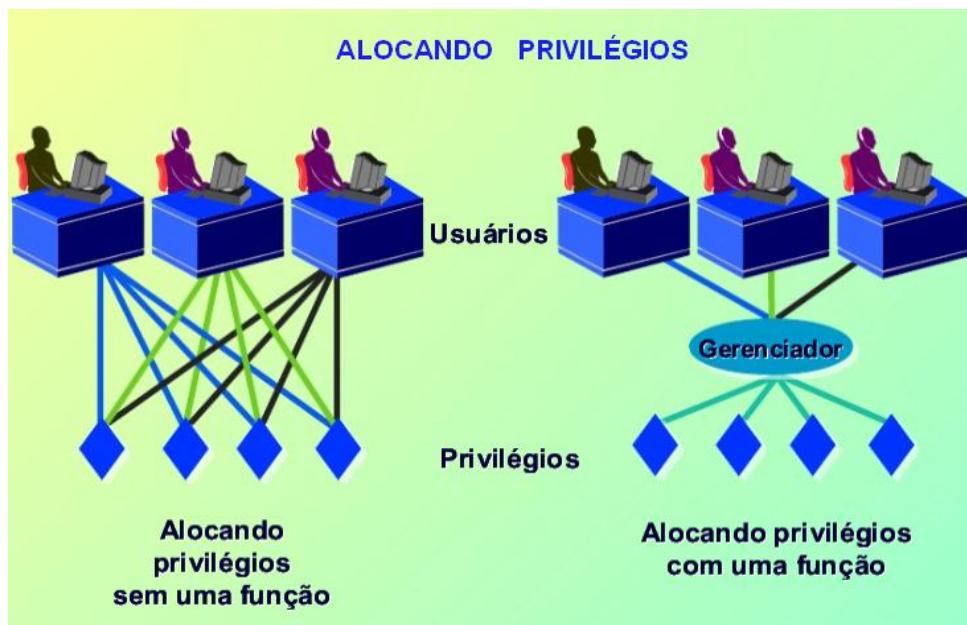


Figura 14.3: Alocando privilégios sem ou com funções

Então, uma *role* visa agrupar privilégios específicos e aplicá-los a todos em um grupo. Ela representa todas as ações que um ou mais usuários podem realizar no banco de dados.

Criando uma *ROLE* (papel)

Para criar uma *role*, veja os passos a seguir:

1. Utilizamos o comando CREATE ROLE da seguinte maneira:

```
CREATE ROLE nome-role;
```

onde:

- ▲ CREATE e ROLE: fazem parte do comando de criação de papéis;
- ▲ *nome_role*: nome identificador da *role* que está sendo criada.

2. É necessário atribuir-lhe todas as ações que representa;

3. Em seguida, essa *role* pode ser atribuída a um ou mais usuários, ou seja, é preciso dar privilégios aos usuários.

Observações:

- ▲ Ao adicionar privilégios a uma *role*, trate-a como se fosse simplesmente um usuário.
- ▲ Todos os usuários a quem a *role* for atribuída receberão os privilégios que foram atribuídos a ela. A qualquer momento é possível atribuir ou revogar privilégios de

uma *role*;

- ▲ As *roles* somente serão úteis se a elas forem associados os privilégios aos quais correspondem (visto anteriormente).

Exemplo:

1. Criando uma role chamada MANAGER:

CREATE ROLE **MANAGER**;

2. Atribuir-lhe todas as ações que representa:

GRANT SELECT, INSERT nome_tabela TO **MANAGER**;

3. *Role* pode ser atribuída a diversos usuários:

GRANT **MANAGER** TO user1, user2, user3, user4;

Eliminando uma *ROLE*

Quando você não precisa mais de sua role, não há razão para mantê-la. Use o comando DROP para removê-la.

Sintaxe:

DROP ROLE *nome-role*;

Observação:

- ▲ Ao removermos uma *role*, ela é automaticamente retirada de todos os usuários a quem tinha sido atribuída.

Início da atividade

Atividade 3 - Atende ao objetivo 2

Considere que o banco de dados LIVROS, contendo as tabelas ASSUNTO, AUTOR e EDITORA será utilizado por diversos usuários que podem ser agrupados segundo dois diferentes perfis: um denominado funcionário (pessoal da biblioteca) e outro denominado visitante (leitor que não possui cadastro). Você deve:

- a) Criar uma *role* em cada perfil de usuário.
- b) Será permitido realizar quaisquer operações sobre os dados das tabelas ASSUNTO,

AUTOR e EDITORA para o perfil funcionário.

- c) Será permitido pesquisar os dados das tabelas ASSUNTO, AUTOR e EDITORA para o perfil visitante.
- d) Considerar os novos funcionários da biblioteca: Ana, João e Camila. Atribuir o acesso completo aos dados da tabelas ASSUNTO, AUTOR e EDITORA.
- e) Após uma reavaliação dos critérios de segurança do banco de dados LIVROS, ficou definido que os funcionários novos não devem ter acesso aos dados da tabela EDITORA.
- f) Considerar que a funcionária Ana não deva mais ter os privilégios definidos para a role FUNCIONARIO.
- g) Novos critérios de segurança serão estabelecidos numa próxima reunião. Então, é preciso remover a role FUNCIONARIO.

Diagramação, inserir 5 linhas para a resposta

Resposta comentada

- a) CREATE ROLE FUNCIONARIO;
CREATE ROLE VISITANTE;
- b) GRANT ALL PRIVILEGES ON ASSUNTO TO FUNCIONARIO;
GRANT ALL PRIVILEGES ON AUTOR TO FUNCIONARIO;
GRANT ALL PRIVILEGES ON EDITORA TO FUNCIONARIO;
- c) GRANT SELECT ASSUNTO TO VISITANTE;
GRANT SELECT AUTOR TO VISITANTE;
GRANT SELECT EDITORA TO VISITANTE;
- d) GRANT FUNCIONÁRIO TO Ana, João, Camila;
- e) REVOKE ALL PRIVILEGIES ON EDITORA FROM FUNCIONARIO;
- f) REVOKE FUNCIONARIO FROM Ana;
- g) DROP ROLE FUNCIONARIO;

Fim da atividade

Conclusão

Para garantir segurança na modelagem de aplicações de banco de dados, deve-se adotar mecanismos de segurança que determinem quais dados serão protegidos e quais usuários terão acesso a eles. Para criar usuários, conceder privilégios e, em contrapartida, retirar um privilégio, utilizam-se os comandos CREATE USER , GRANT e REVOKE, respectivamente.

Uma ROLE (papel) visa agrupar privilégios específicos e aplicá-los a todos em um grupo, facilitando a inclusão e/ou remoção de privilégios.

Finalmente, tabelas GRANT do MySQL são o coração do seu sistema de segurança, além dos privilégios REVOKE e DROP USER.

Resumo

O SGBD possui privilégios específicos como:

- Conectar-se a uma instância do SGBD ou a uma de suas bases de dados;
- Criar e destruir objetos do banco de dados como tabelas, índices, visões, gatilhos e procedimentos armazenados, entre outros;
- Consultar, atualizar, incluir e excluir dados em tabelas (e em visões, para os gerenciadores que suportam a realização dessas operações nas mesmas);
- Executar procedimentos e funções armazenados;
- Alterar as características do sistema e de suas bases de dados, tais como parâmetros de inicialização;
- Realizar operações de geração de cópias de segurança (*backup*) e recuperação de tais cópias (*restore*).

Direito de sistema

Trata-se de como obter acesso ao banco de dados. Mas, antes disso precisamos aprender como criar e remover usuários. Quando o usuário for criado, o DBA poderá

conceder privilégios de sistema específicos para ele.

- ▲ Por padrão, o primeiro usuário – o usuário root (raiz) – tem controle completo sobre qualquer coisa no banco de dados. Isto é importante porque o usuário raiz precisa estar apto a criar contas de usuários para todos os outros usuários.

Atribuir privilégios de sistema

O DBA permite ao usuário controlar todos os objetos do banco de dados. É um direito que deve ser concedido com muito crédito para evitar problemas ao banco de dados. Recomenda-se que apenas o administrador do banco de dados o possua.

Sintaxe:

```
GRANT <tipo de privilégios> ON {NomeBanco.tabela | NomeBanco.* | *.*}
TO <nome usuario> @ <dominio> [IDENTIFY BY <senha>] [, nome
usuario2 [IDENTIFY BY <senha2>] ...
[WITH GRANT OPTION]
```

onde:

- ▲ nome_usuario: nome do usuário para quem o privilégio será concedido.
- ▲ GRANT OPTION: permite ao usuário repassar os seus privilégios.

Revogar privilégios de sistema

Após o usuário ter recebido o privilégio para executar uma ação, ele irá manter esse privilégio até que ele seja explicitamente revogado. Para revogar um ou mais privilégios de um usuário, deve-se utilizar o comando REVOKE.

Sintaxe:

```
REVOKE [GRANT OPTION FOR] <Privilégios> ON objeto FROM
nome_usuario [CASCADE CONSTRAINTS];
```

onde:

- ▲ REVOKE e FROM: Fazem parte do comando;
- ▲ ON objeto: Refere-se ao nome do banco de dados ou tabela em questão;

- ▲ GRANT OPTION FOR: É opcional. Indica que a habilidade de conceder o privilégio em questão para outros usuários será revogada;
- ▲ nome_usuário: Nome de um ou mais usuários ou papéis para quem o privilégio será revogado, separados por vírgula.

Conceder grupos de direitos de acesso

Atribuir privilégios referentes a cada um dos objetos para cada usuário de forma individual pode ser uma tarefa bastante trabalhosa. Além disso, à medida que aumenta o número de tabelas ou outros objetos sobre os quais cada usuário deve ter privilégios, aumenta também a probabilidade de que, por erro ou esquecimento, um privilégio necessário não seja atribuído, podendo em alguns casos causar graves problemas. Para facilitar as tarefas de administração do banco de dados relacionadas com a atribuição e revogação de privilégios e reduzir o número de erros nessas operações, foi definido o conceito de *roles* (funções).

Uma *role* visa agrupar privilégios específicos e aplicá-los a todos em um grupo. Ela representa todas as ações que um ou mais usuários podem realizar no banco de dados.

Criando uma *ROLE*

Criar uma *role*: utilizamos o comando CREATE ROLE.

Sintaxe:

`CREATE ROLE nome-role;`

onde:

- ▲ CREATE e ROLE: fazem parte do comando de criação de papéis;
- ▲ *nome_role*: nome identificador da *role* que está sendo criada.

Eliminando uma *ROLE*

Quando você não mais precisa de sua *role*, não há razão para mantê-la. Use o comando DROP para se ver livre dela.

Sintaxe:

```
DROP ROLE nome-role;
```

Observação:

- ▲ Ao remover uma *role*, ela é automaticamente retirada de todos os usuários a quem tinha sido atribuída.

Referências bibliográficas

- CARVALHO C. R. *SQL - Guia Prático*. 2^a ed. Rio de Janeiro: Brasport, 2006.
- DATE C. J. *Introdução a Sistemas de Bancos de Dados*. 8^a ed. americana. Rio de Janeiro: Elsevier, 2003.
- ELMASRI, R.; NAVATHE S. *Sistemas de Banco de Dados*. 5^a ed. São Paulo: Pearson Addison Wesley, 2009.
- HEUSER, C. A. *Projeto de Banco de Dados*. 4^a ed. Porto Alegre: Bookman, 2009.
- JOBSTRAIBIZER, F. *Criação de banco de dados com MySQL*. São Paulo: Digerati Books, 2010.
- SETZER, V. W.; CORRÊA DA SILVA, F. S. *Bancos de Dados*. São Paulo: Edgard Blucher, 2005.
- SILBERSCHATZ, A.; KORTH, H. *Sistema de Banco de Dados*. 3^a ed. São Paulo: Pearson, 2008.