

O guia amigo do seu cérebro

# Use a Cabeça!

(Head First)

# SQL



Ajude Greg a melhorar seus relacionamentos com seus dados, com este livro ganhador do prêmio JOLT 2008



Nunca mais você irá posicionar de forma errada suas chaves primária e estrangeira



Finalmente sinta-se capaz de explicar coisas normais



Carregue importantes conceitos sobre query diretamente para dentro do seu cérebro



Evite os embaraçosos cenários ALTER



Coloque seu conhecimento sobre SQL em cheque com dezenas de exercícios

O'REILLY®

A  
ALTA BOOKS  
EDITORIA

Lynn Beighley

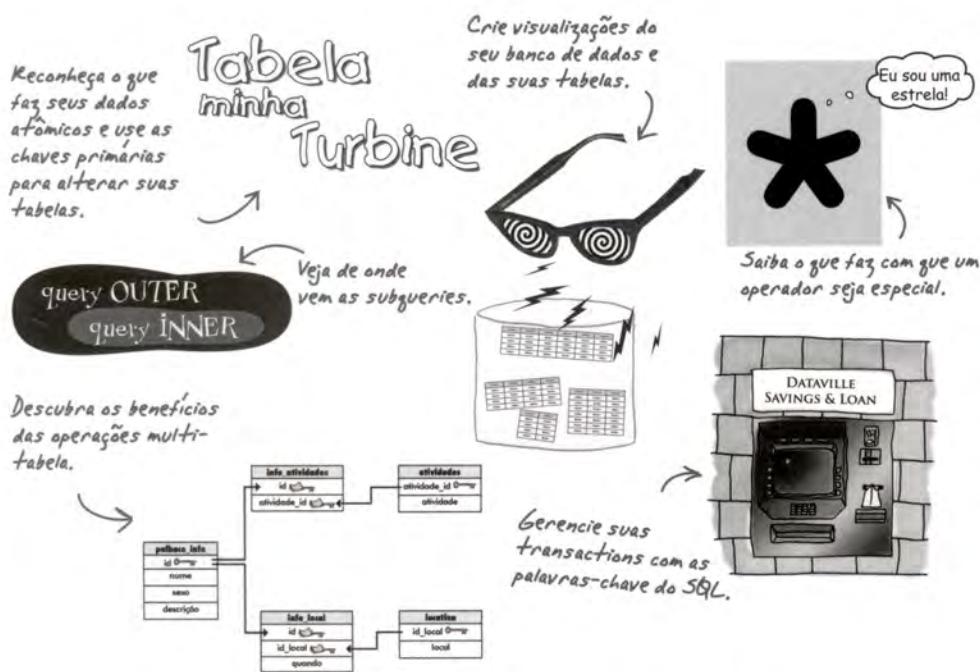
# Use a Cabeça! SQL

(Head First)

Banco de Dados/SQL

## O que você irá aprender com este livro?

No mundo de hoje, dados é poder, mas o verdadeiro segredo do sucesso é ter poder sobre seus dados. *Use a Cabeça SQL* leva você ao coração da linguagem SQL, da sintaxe básica das queries, usando INSERT e SELECT, à dureza da manipulação do banco de dados com subqueries, joins e transactions. A medida que você avança na leitura, entenderá efetiva e eficientemente o projeto e a criação de banco de dados, utilizando queries, normatização e joining. Você será então o verdadeiro mestre dos seus dados!



## Por que este livro parece tão diferente?

Nós acreditamos que seu tempo é muito valioso para ser desperdiçado. Tendo como base a última pesquisa em neurobiologia, ciência cognitiva e teoria do aprendizado, *Use a Cabeça SQL* tem um visual rico, projetado na forma como seu cérebro funciona; não se trata de uma abordagem pesada que faz com que você caia em sono profundo.

"Este livro não torna o SQL mais fácil, mas o torna desafiador, interessante e divertido. Isso até responde à pergunta 'Como ensinar queries não relacionadas sem perder a vontade de viver?' Esta é a forma correta de aprender - é fácil, é vibrante e tem uma aparência incrível."

— Andrew Cumming, Autor de SQL Hacks, [sqlzoo.net](http://sqlzoo.net)

"Existem aqueles livros que você compra, livros que você guarda, livros que você deixa na sua mesa, e graças à equipe do Use a Cabeça, há uma última categoria, os livros da série Use a Cabeça.

São aqueles livros com as páginas cheias de orelhas, usados e carregados para todos os lados. Use a Cabeça SQL está no topo da minha pilha".

— Bill Sawyer, Gerente de currículos, Oracle

ISBN 978-85-7608-210-1



A  
ALTA BOOKS  
E D I T O R A  
Rua Viúva Claudio, 291 - Jacaré  
Rio de Janeiro CEP 20970-031  
Tel: 21 3278-8069/ Fax 3277-1253  
[www.altabooks.com.br](http://www.altabooks.com.br)  
[altabooks@altabooks.com.br](mailto:altabooks@altabooks.com.br)

O'REILLY®

## **Elogios a outros livros da série Use a Cabeça**

“Eu acabei de ler Use a Cabeça OOA&D e amei o livro! O que mais gostei neste livro foi o foco no porquê de utilizarmos OOA&D – para fazer ótimos programas.”

— **Kyle Brown, Engenheiro conceituado, IBM.**

“Eu decorei Use a Cabeça HTML com CSS & XHTML - ele ensina tudo o que você precisa saber com um formato divertido de citações”.

— **Sally Appling, Designer e Artista da UI, <http://sally.com>**

## **Elogios para a abordagem de Use a Cabeça**

“É rápido, irreverente, divertido e empolgante. Cuidado! Pode ser que você realmente aprenda alguma coisa”.

— **Ken Arnold, Ex-Engenheiro Chefe da Sun Microsystems  
Co-autor (junto com James Gosling de Java), *The Java Programming Language***

“Eu senti como se 1000 quilos de livros acabassem de ser levantados de cima da minha cabeça.”

— **Ward Cunningham, Inventor da Wiki e fundador do Grupo Hillside.**

“Este livro é o mais próximo da perfeição, tendo em vista a maneira como combina técnica e facilidade de leitura. Ele fala com autoridade e a leitura é linda”.

— **David Galernter, Professor de Ciências da Computação, Universidade Yale.**

“Este é o tom exato para o guru da programação nerd-extrovertido, descolado-casual que existe em todos nós. É a referência certa para as estratégias práticas de desenvolvimento. Acelera meu cérebro sem ter que rastejar pelo discurso cansado e enferrujado da aula de um professor”.

— **Travis Kalanick, Fundador da Scour and Red Swoosh e membro da MIT TR100**

“A combinação do humor, ilustração, barras laterais e redundância, com uma abordagem lógica para apresentar os comandos básicos e exemplos substanciais de como usá-los fará, esperançosamente, com que os leitores se prendam de tal forma que eles nem percebam que estão aprendendo por estarem se divertindo tanto”.

— **Stephen Chapman, Fellgall.com**

## Elogios de Especialistas para Use a Cabeça SQL

"Existem aqueles livros que você compra, livros que você guarda, livros que você deixa na sua mesa, e graças a equipe do Use a cabeça, há uma penúltima categoria, os livros da série Use a Cabeça. Eles são os livros com orelhas, usados e carregados para todos os lados. Use a Cabeça SQL está no topo da minha pilha. Puxa, até a versão PDF que guardo para revisão está gasto de tanto uso".

— Bill Sawyer, Gerente de currículos, Oracle

"Este livro não é sobre tornar SQL fácil, mas torná-lo desafiador, torná-lo interessante, torná-lo divertido. Isso até responde àquela pergunta antiga 'Como ensinar consultas não relacionadas sem perder a vontade de viver?' Esta é a forma correta de aprender – é fácil, é vibrante, e tem uma aparência incrível."

— Andrew Cumming, Autor de *SQL Hacks*, mantenedor do [sqlzoo.net](http://sqlzoo.net)

"Fascinante! Quero dizer, SQL é uma linguagem de computador, certo? Logo, livros sobre SQL deveriam ser escritos para computadores, não acha? Use a Cabeça é obviamente escrito para seres humanos! Como é que isto é possível?"

— Dan Tow, autor de *SQL Tuning*

## A série Use a Cabeça



Use a Cabeça Java 2a Ed.  
Kathy Sierra e Bert Bates  
496 pgs  
Formato 21x28cm  
ISBN 978-85-7608-173-9



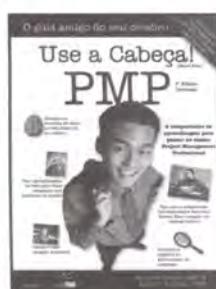
Use a Cabeça Análise e Projeto Orientado ao Objeto  
McLaughlin, Pollice & West  
472 pgs  
Formato 21x28cm  
ISBN 978-85-7608-145-6



Use a Cabeça Padrões de Projetos 2a Ed.  
Freeman e Freeman  
496 pgs  
Formato 21x28cm  
ISBN 978-85-7608-174-6



Use a Cabeça Servlets & JSP  
Basham, Sierra e Bates  
616 pgs  
Formato 21x28cm  
ISBN 85-7608-085-0



Use a Cabeça PMP, 2<sup>a</sup> Ed.  
Greene e Stellman  
624 pgs  
Formato 21x28cm  
ISBN 978-85-7608-309-2



Use a Cabeça (Iniciação Rápida) Ajax 2a Ed.  
Brett McLaughlin  
344 pgs  
Formato 21x28cm  
ISBN 978-85-7608-193-7



Use a Cabeça HTML com CSS & XHTML 2a Ed.  
Freeman e Freeman  
504 pgs  
Formato 21x28cm  
ISBN 85-7608-105-9

## E a Lista Continua...

- Use a Cabeça Estatística
- Use a Cabeça EJB
- Use a Cabeça C#
- Use a Cabeça JavaScript
- Use a Cabeça Física
- Use a Cabeça Programando
- Use a Cabeça Desenvolvimento de Software
- Use a Cabeça Ajax
- Use a Cabeça Servlets & JSP

## **Elogios a outros livros da série Use a Cabeça**

“A admirável clareza, humor e doses substanciais de esperteza tornam este o tipo de livro que ajuda mesmo aqueles que não são programadores a pensar sobre solução de problemas”.

**- Cory Doctorow, Co-editor de Boing Boing**

**Autor de Down and out in the Magic Kingdom (Triste e Cabisbaixo no Reino da Fantasia) e Someone Comes to Town, Someone Leaves Town (Alguém Vem até a Cidade, Outro Alguém Sai)**

“Se você pensou que Ajax era ciência de foguetes, este livro é pra você. Head Rush Ajax o coloca diversas experiências dinâmicas e tocantes dentro da realidade de cada programador em web”.

**- Jesse James Garret, Caminho Adaptado.**

“Eu recebi o livro ontem e comecei a lê-lo... e não consegui parar. Este é, com certeza, um livro muito legal. É divertido, mas eles abordam diversas áreas e vão direto ao assunto. Estou realmente impressionado”.

**- Eric Gamma, Engenheiro conceituado e co-autor de Padrões de Design.**

“Use a Cabeça Padrões de Design conseguiu uma criar uma mistura de diversão, gargalhadas, profundidade de conteúdo e grandes conselhos práticos em uma leitura divertida e instigante. Quiçá você seja novo em padrões de design ou as tem usado por anos, com certeza, aprenderá alguma coisa se visitar Vila do Objeto”.

**- Richard Helm, Co-autor de Padrões de design.**

“Um dos mais livros mais inteligentes e engraçados sobre design de software que eu já li”.

**- Aaron LaBerge, Vice-Presidente de Tecnologia, ESPN.com**

### 3.3. Definitions

Let  $\mathcal{G}$  be a graph with vertex set  $V(\mathcal{G})$  and edge set  $E(\mathcal{G})$ . A *sh-path* in  $\mathcal{G}$  is a path  $P$  such that  $(V(P), E(P))$  is a subgraph of  $\mathcal{G}$  and  $(V(P), E(P))$  is a sh-set. A *sh-cycle* in  $\mathcal{G}$  is a cycle  $C$  such that  $(V(C), E(C))$  is a subgraph of  $\mathcal{G}$  and  $(V(C), E(C))$  is a sh-set.

### 3.4. Proof of Theorem 1

#### 3.4.1. Definitions

##### 3.4.1.1. Sh-graphs

A *sh-graph* is a graph  $\mathcal{G}$  such that every sh-set in  $\mathcal{G}$  is a sh-cycle or a sh-path. In other words, a sh-graph is a graph in which every sh-set is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.2. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.3. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.4. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.5. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.6. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.7. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

#### 3.4.1.8. Sh-graphs and sh-graphs

Let  $\mathcal{G}$  be a graph. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1. We say that  $\mathcal{G}$  is a *sh-graph* if and only if every sh-set in  $\mathcal{G}$  is a sh-set in the sense of Definition 3.2.1.

# Use a Cabeça SQL

Não seria um sonho se existisse um livro que pudesse me ensinar SQL sem que me desse vontade de querer mudar para uma ilha deserta do Pacífico onde não existam bancos de dados? Provavelmente isto não é nada mais que uma fantasia..."

Lynn Beighley



## **Use a Cabeça! (Head First) SQL**

Do original **Head First SQL** Copyright © 2008 da Editora Alta Books Ltda.

Authorized translation from English language edition, entitled **Head First SQL**, ISBN 0-596-52684-9, by Lyn Beighley, published by O'Reilly Media, Inc. Copyright © 2007 by O'Reilly Media, Inc..

PORTUGUESE language edition published by Editora Alta Books, Copyright © 2008 by Editora Alta Books.

Todos os direitos reservados e protegidos pela Lei 5988 de 14/12/73. Nenhuma parte deste livro, sem autorização prévia por escrito da editora, poderá ser reproduzida ou transmitida sejam quais forem os meios empregados: eletrônico, mecânico, fotográfico, gravação ou quaisquer outros. Todo o esforço foi feito para fornecer a mais completa e adequada informação, contudo a editora e o(s) autor(es) não assumem responsabilidade pelos resultados e usos da informação fornecida. Recomendamos aos leitores testar a informação, bem como tomar todos os cuidados necessários (como o backup), antes da efetiva utilização. Este livro não contém CD-ROM, disquete ou qualquer outra mídia.

**Erratas e atualizações:** Sempre nos esforçamos para entregar a você, leitor, um livro livre de erros técnicos ou de conteúdo; porém, nem sempre isso é conseguido, seja por motivo de alteração de software, interpretação ou mesmo quando alguns deslizes constam na versão original de alguns livros que traduzimos. Sendo assim, criamos em nosso site, [www.altabooks.com.br](http://www.altabooks.com.br), a seção Erratas, onde relataremos, com a devida correção, qualquer erro encontrado em nossos livros.

**Avisos e Renúncia de Direitos:** Este livro é vendido como está, sem garantia de qualquer tipo, seja expressa ou implícita.

**Marcas Registradas:** Todos os termos mencionados e reconhecidos como Marca Registrada e/ou comercial são de responsabilidade de seus proprietários. A Editora informa não estar associada a nenhum produto e/ou fornecedor apresentado no livro. No decorrer da obra, imagens, nomes de produtos e fabricantes podem ter sido utilizados, e desde já a Editora informa que o uso é apenas ilustrativo e/ou educativo, não visando ao lucro, favorecimento ou desmerecimento do produto/fabricante.

**Produção Editorial:** Editora Alta Books

**Coordenação Editorial:** Thalita Aragão Ramalho

**Tradução:** Leopoldino Machado

**Revisão:** Romulo Leite da Cunha

**Revisão Técnica:** Anderson Bastos e Ivan Mecenas

**Diagramação:** Renan Salgado

**2ª Reimpressão 2010**

Impresso no Brasil

O código de propriedade intelectual de 1º de Julho de 1992 proíbe expressamente o uso coletivo sem autorização dos detentores do direito autoral da obra, bem como a cópia ilegal do original. Esta prática generalizada nos estabelecimentos de ensino, provoca uma brutal baixa nas vendas dos livros a ponto de impossibilitar os autores de criarem novas obras.



Rua Viáva Claudio, 291 - Jacaré  
Rio de Janeiro - RJ  
CEP: 20020-100  
Tel: 21 3278-8069 / Fax: 3277-1253  
[www.altabooks.com.br](http://www.altabooks.com.br),  
e-mail: [altabooks@altabooks.com.br](mailto:altabooks@altabooks.com.br)

Para o nosso mundo, em um mar de dados.  
E para você, que quer reinar sobre este mar.

SO  
TYPE IN  
FORMAT



BY

JOSEPH

WILSON

ILLUSTRATION

BY

JOSEPH

WILSON

DESIGN

BY

JOSEPH

WILSON

PRINTED

BY

JOSEPH

WILSON

PUBLISHING

BY

JOSEPH

WILSON

PRINTED

BY

JOSEPH

WILSON

PUBLISHING

BY

## Autora de Use a Cabeça SQL



Lynn Beighley

**Lynn** é uma escritora de ficção em um corpo de escritora de livro técnico. Após descobrir que escrever livros técnicos rende um bom dinheiro, ela aprendeu a aceitar este fato e a gostar disso.

Após voltar para a escola e conseguir seu mestrado em ciências da computação, ela trabalhou para as siglas NRL e LANL. Então ela conheceu o Flash e escreveu seu primeiro best-seller.

Uma vítima da hora errada, ela mudou-se do Vale de Silício um pouco antes da quebra-deira. Ela passou vários anos trabalhando para o Yahoo! e escrevendo outros livros e cursos de treinamento. Finalmente se entregando ao seu lado criativo de escritora, ela mudou-se para Nova York para conseguir um MFA (formação para escritores) em escrita criativa.

Sua primeira tese em estilo Use a Cabeça foi exposta em uma sala lotada de professores e colegas estudantes. A tese foi extremamente bem recebida e Lynn concluiu seu primeiro mestrado, terminou o livro Use a Cabeça SQL e não vê a hora de começar a escrever seu próximo livro.

Lynn ama viajar, cozinhar e inventar estórias bem elaboradas sobre pessoas desconhecidas. Ela tem um pouco de medo de palhaços.



↑  
A vista da janela  
de Lynn.

# Índice (Sumário)

Intro	xxv
1 Dados e Tabelas: Um lugar para cada coisa	1
2 O comando SELECT: Abençoado restaurador de dados	47
3 DELETE e UPDATE: Uma mudança fará bem a você	97
4 Projetos de Tabelas inteligentes: Por que ser normal?	133
5 ALTER: Reescrivendo o passado	159
6 SELECT avançado: Ver os seus dados com novos olhos	187
7 Projeto de Banco de dados multi-tabelas: Povoando sua tabela	227
8 Conexões e Operações de multi-tabelas: Não podemos todos nos entender?	277
9 Subconsultas: Consultas dentro de Consultas	307
10 Conexões externas, intraconexões e uniões: Novas manobras	339
11 Constraints, views e transactions: Cozinhar demais pode estragar o banco de dados	369
12 Segurança: protegendo suas riquezas	401

## Intro

**Seu cérebro em SQL.** Aqui está você, tentando aprender alguma coisa, enquanto aqui seu cérebro está fazendo um favor certificando-se de que o aprendizado não se fixe. Seu cérebro está pensando: "Melhor deixar mais espaço para coisas mais importantes, tipo qual animal selvagem deve-se evitar e em quais situações esquiar nu é uma má idéia. Então como você despista seu cérebro fazendo ele pensar que sua vida depende de saber SQL?"

Para quem é esse livro?	xxvi
Sabemos o que você está pensando	xxvi
Metacognição: Pensando sobre o pensar	xxix
Curve o seu cérebro à submissão	xxxI
Leia-me	xxxii
O time de colaboração técnica	xxxiii
Agradecimentos	xxxiv

dados e tabelas

## Um lugar para cada coisa

# 1

### Você simplesmente não odeia perder as coisas?

Seja a chave do seu carro, ou aquele cupom de 25% de desconto da sua loja de departamentos preferida, ou os dados de seus aplicativos, não há nada pior que não poder guardar aquilo que você precisa... quando você precisa. E quando o assunto é seus aplicativos, não há lugar melhor para armazenar informações importantes do que uma **tabela**. Então vire a página, entre, e viaje pelo mundo dos **bancos de dados relacionais**.

Definindo seus dados	2
Olhe para seus dados em categorias	6
O que fica em um banco de dados?	7
Seu banco de dados visto através de uma visão raio-x...	9
Bancos de dados contêm dados interligados	11
Visão de perto: Tabelas	12
Tome o comando!	16
Preparando a tabela: o comando CREATE TABLE	18
Criando uma tabela um pouco mais complicada	19
Veja como é fácil escrever em SQL	20
Finalmente, criando a tabela meus_contatos	21
Sua tabela está pronta	22
Perambulando por aí	23
Sua tabela DESCrita	26
Você não pode reciar uma tabela ou banco de dados já existente!	28
Fora com a tabela antiga, pra dentro com a nova	29
Para inserir dados na tabela, você usará o comando INSERT	31
Criando o comando INSERT	33
Variações para o comando INSERT	37
Colunas sem valores	38
Espie sua tabela com o comando SELECT	39
SQL exposto	40
Controlando seu NULL interior	40
NOT NULL aparece em DESC	42
Preencha os espaços em branco com DEFAULT	43
Sua caixa de ferramenta SQL	44



## O comando SELECT

### **Abençoado restaurador de dados**

# 2

**Fala sério, é melhor dar do que restaurar?** Quando o assunto é bancos de dados, as chances são de que você terá que **dar os seus dados** na mesma freqüência que as insere. É aí que este capítulo entra: você conhecerá o poderoso comando **SELECT** e aprender como **ganhar acesso àquela importante** informação que tem inserido nas tabelas. E ainda vai aprender como usar **WHERE, AND e OR** para acessar suas tabelas e ainda evitar a exibição de dados de que *não precisa*.

Marcar ou não marcar um encontro?	48
Um SELECT melhor	50
Que * é esta?	51
Como fazer consultas nos seus tipos de dados	56
Mais problemas de pontuação	56
Aspas simples incompatíveis	57
Aspas simples são caracteres especiais	58
INSERT (Insira) dados com aspas simples nele	59
SELECT (selecionar) colunas específicas para limitar os resultados	63
SELECT (selecione) colunas específicas para resultados mais rápidos	63
Combinando suas consultas	67
Encontrando valores numéricos	70
Operadores de comparação sem <del>problemas</del>	72
Encontrando dados numéricos com operadores de comparação	74
Amarrando os dados em texto com Operadores de Comparação	77
Ser OR (ou) não ser	78
A diferença entre AND e OR	80
Usando IS NULL para encontrar NULL	82
Economizando tempo com uma simples palavra-chave: LIKE	84
O chamado do Coringa	84
Selecionando alcance usando AND e operadores de comparação	87
Bem BETWEEN (entre) nós... há um jeito melhor	87
Depois dos encontros, você está IN...	89
...ou você está NOT IN	89
Mais NOT	90
Sua caixa de ferramenta SQL	94



## 3

## DELETE e UPDATE

**Uma mudança fará bem a você**

**Você muda de opinião constantemente? Agora está tudo bem!** Com os comandos que você está prestes a conhecer – DELETE e UPDATE – você não mais precisará ficar preso a uma decisão que tomou há seis meses atrás, quando inseriu dados dizendo que calça boca-de-sino logo estaria de volta à moda. Com UPDATE, você **pode mudar os dados**, e DELETE permite **eliminar os dados** que não precisar mais. Mas não estamos apenas lhe dando as ferramentas; neste capítulo, você aprenderá a ser mais seletivo com seus novos poderes e a evitar que suma com arquivos que realmente precise.

Palhaços são assustadores?	98
Rastreador de palhaços	99
Os palhaços estão em movimento	100
Como os nossos dados de palhaços são inseridos?	104
Bonzo, nós temos um problema	105
Eliminando um registro com DELETE	106
Usando nossa nova instrução DELETE	107
Regras do DELETE	108
Os dois passos do INSERT e DELETE	111
Cuidado com o seu DELETE	116
O problema com DELETE impreciso	119
Altere seus dados com UPDATE	120
Regras do UPDATE	121
UPDATE é o novo INSERT-DELETE	122
UPDATE em ação	123
Atualizando os movimentos dos palhaços	124
UPDATE (atualize) seus preços	128
Tudo o que precisamos é um UPDATE	130
Sua caixa de ferramenta SQL	131



## Projetos de Tabelas Inteligentes

### Por que ser normal?

# 4

**Você tem criado tabelas sem dar muita atenção a elas.** Tudo bem quanto a isso, elas funcionam. Você pode usar o SELECT, INSERT, DELETE e UPDATE nelas. Mas, à medida que você insere mais dados, começa a ver coisas que vai desejar ter feito para que a sua cláusula WHERE fosse mais simples. Você precisa é fazer suas tabelas mais normais.

Duas tabelas de pescadores	134
Uma tabela tem tudo a ver com relacionamentos.	136
Dados atômicos	139
Dados atômicos e suas tabelas	140
Regras do dados atômicos	141
Razões para ser normal	143
O benefício de tabelas normais	144
Palhaços não são normais	145
Metade do caminho para 1NF	146
Regras da CHAVE PRIMÁRIA	146
Ficando NORMAL	148
Consertando a tabela do Greg	148
O comando CREATE TABLE que nós escrevemos	149
Mostrar-me o <del>diretório</del>	150
Comando para economia de tempo	150
O CREATE TABLE com uma CHAVE PRIMÁRIA	151
1, 2, 3 ... auto incrementando	152
Adicionando uma CHAVE PRIMÁRIA a uma tabela existente	155
ALTER TABLE e adicionar uma PRIMARY KEY	156
Sua caixa de ferramenta SQL	157



## ALTER

## Reescrevendo o passado

## 5

**Você gostaria corrigir os erros do seu passado?** Bem, agora é sua chance. Apenas usando o comando ALTER, você pode aplicar a todas as lições que tem aprendido sobre as tabelas que criou há dias, meses e até anos atrás. Melhor ainda, você pode fazer isso sem alterar seus dados. No momento que tiver passado por aqui, você saberá o que o **normal** realmente significa, e estará apto a aplicar isso em todas as suas tabelas, passadas ou presentes.

Nós precisamos fazer algumas mudanças	160
Alterações de tabelas	164
Reconstrução total de tabelas	165
Renomeando a tabela	165
Nós precisamos fazer alguns planejamentos	167
Reinstrumentalizando nossas colunas	167
Mudanças estruturais	168
ALTER e CHANGE	168
Altere duas colunas com apenas um comando SQL	169
Rápido! Largue esta coluna!	172
Um olhar de perto para a coluna não-atômica “local”	177
Procure por padrões	178
Algumas funções de texto muito úteis	178
Utilize uma coluna atual para preencher uma nova coluna	181
Como funciona nosso combo UPDATE e SET?	182
Sua caixa de ferramenta SQL	184

# Turbine minha Tabela

É hora de você transformar sua tabela velha e cansada em um imã de dados e a levar a um nível de turbinação de tabela que você nunca pensou que existisse.



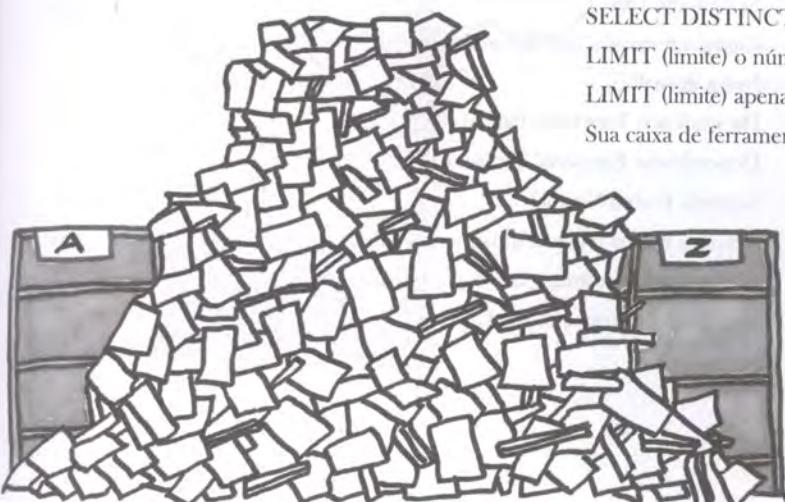
## SELECT avançado

### Vendo sua tabela com novos olhos

# 6

**É hora de incrementar a sua caixa de ferramentas com um pouco de finesse.** Você já sabe como utilizar as cláusulas WHERE e o comando SELECT. Mas, às vezes, precisa de mais **exatidão** que o SELECT e o PROVIDE oferece. Neste capítulo, você aprenderá como **ordenar e agrupar** seus dados, bem como **realizar operações matemáticas** nos seus resultados.

Dataville Video está se reorganizando	188
Problemas com nossa tabela atual	189
Combinando dados existentes	190
Povoando sua nova coluna	190
UPDATE com uma expressão CASE	193
Parece que temos um problema	195
As tabelas podem ficar bem bagunçadas	199
Nós precisamos de uma maneira de organizar os dados que selecionamos	200
Tente um simples ORDER BY	203
ORDER (Ordene) uma só coluna	204
ORDER com duas colunas	207
ORDER com colunas múltiplas	207
Uma tabela_filme organizada	209
Inverta o comando ORDER com DESC	210
O problema da líder de vendas de biscoitos da Bandeirantes	212
SUM pode somá-las para nós.	214
SUM (some) todas de uma vez com GROUP BY	215
AVG combinada com GROUP BY	216
MIN e MAX	217
COUNT (conte) os dias	217
SELECT DISTINCT (selecione diferentes) valores	219
LIMIT (limite) o número de resultados	221
LIMIT (limite) apenas para a segunda colocada	222
Sua caixa de ferramenta SQL	224



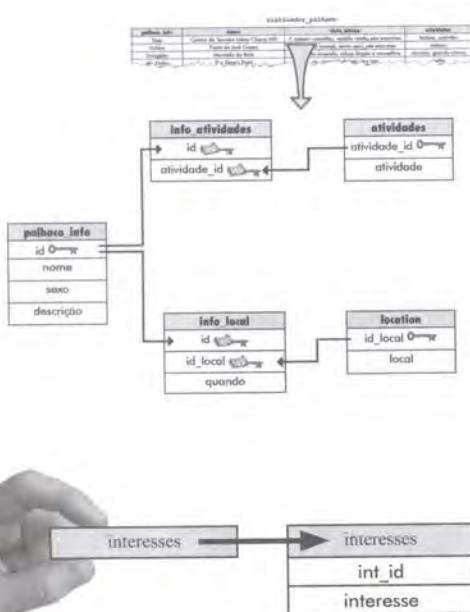
## Projeto de bancos de dados multi-tabelas

### Povoando sua tabela

7

#### Às vezes sua tabela única já não é grande o bastante.

Seus dados se tornaram mais complexos, e aquela tabela única que você tem utilizado, já não vai mais servir. Sua tabela é cheia de dados redundantes, gastando espaço e reduzindo a velocidade de suas consultas. Você já foi o mais longe que podia com uma tabela só. Há um mundo grande lá fora e, às vezes, é preciso mais de uma tabela para conter dados, controlar e, principalmente, ser o mestre de seu próprio banco de dados.



Encontrando uma namorada para Nigel	228
Tudo foi em vão... ...Mas espere	237
Pense além da tabela única	238
O banco de dados rastreador de palhaços de múltiplas tabelas	239
O esquema do banco de dados rastreador_palhaco	240
Como ir de uma tabela para duas	241
Conectando suas tabelas	243
Restringindo sua chave externa	246
Por que se importar com chaves estrangeira?	246
CREATE uma tabela com uma CHAVE ESTRANGEIRA	247
Relacionamentos entre tabelas	249
Padrões de dados: um-a-um	249
Padrões de dados: quando usar tabelas um-a-um	250
Padrões de dados: um-para-muitos	250
Padrões de dados: chegando ao muitos-para-muitos	251
Padrões de dados: nós precisamos de uma junction table	253
Padrões de dados: muitos-para-muitos	254
Finalmente 1NF	258
Chaves compostas utilizam múltiplas colunas	258
Notas de atalho	260
Dependência Funcional Parcial	262
Dependência Funcional Transitória	262
Segunda Forma Normal	265
Terceira Forma Normal (finalmente)	270
E, então, Regis (e gregs_list) viveram felizes para sempre	272
Sua caixa de ferramentas SQL	273

## Coneções e operações de multi-tabelas

### **Não podemos todos nos entender?**

# 8

**Bem-vindos a um mundo multi-tabela.** É ótimo ter mais de uma tabela em seu banco de dados, mas você terá algumas novas técnicas e ferramentas para trabalhar com elas. Com as múltiplas tabelas vem também confusão, então precisará de aliases (apelidos) para manter sua tabela correta. E conexões ajudam você a juntar tabelas, para que possa pegar todos os dados que espalhou. Prepare-se, é hora de tomar o controle do seu banco de dados novamente.

Ainda repetindo, repetindo...	278
Pré-povoando suas tabelas	279
Nós tocamos o samba “esta tabela não é fácil de normalizar”	281
Os interesses especiais (coluna)	281
Mantenha-se interessado	282
UPDATE todos seus interesses	283
Pegando todos os interesses	284
Muitos caminhos para um só lugar	285
CREATE, SELECT, e INSERT a (quase) o mesmo tempo	285
CREATE, SELECT e INSERT ao mesmo tempo.	286
O que há com este AS?	286
Pseudônimos de coluna	287
Pseudônimos de tabelas, quem precisa deles?	288
Tudo o que você gostaria de saber sobre conexões internas	288
Conexão cartesiana	289
Liberando sua conexão interna	293
A conexão interna em ação: a equijoin	293
A conexão interna em ação: a não-equijoin	296
A última conexão interna: a conexão natural	298
Consultas conectadas?	303
Pseudônimos de Colunas e Tabelas Revelados	304
Sua caixa de ferramentas SQL	305



...e foi daí  
que tabelas com  
pequenos resultados  
se originaram.

## Subconsultas

# 9

## Consultas dentro de consultas

**Sim, Jack, eu gostaria de uma pergunta de duas partes, por favor.** Conexões são ótimas, mas às vezes você precisa perguntar ao seu banco de dados mais que uma pergunta. Ou pegar o resultado de uma consulta e usá-lo como entrada para outra consulta. É aí que as subconsultas entram. Elas irão ajudar a evitar dados duplicados, fazer suas consultas mais dinâmicas, e ainda fazer com que você tenha acesso aos concertos caríssimos depois da festa. (Bem, na verdade não, mas duas de três coisas já não é nada mal!)

Greg entra no negócio de recrutamento laboral	308
A gregs_list fica com mais tabelas	309
Greg utiliza uma conexão interior	310
Mas ele quer tentar algumas outras consultas	311
Subconsultas	312
Nós combinamos as duas em uma consulta com uma subconsulta	314
Como se uma consulta já não fosse o bastante, conheça a subconsulta	315
Uma subconsulta em ação	315
Regras para subconsultas	317
Regras para subconsultas	318
Um acompanhamento na construção de uma subconsulta	319
Uma subconsulta como uma coluna no comando SELECT	322
Outro exemplo: Subconsulta com uma conexão natural	323
Uma consulta não-correlacionada	324
Uma subconsulta não-correlacionada com múltiplos valores: IN, NOT IN	328
Subconsultas correlacionadas	332
Uma subconsulta correlacionada (útil) com NOT EXISTS	332
EXISTS e NOT EXISTS	333
A empresa de Greg para serviços de recrutamento está aberta para negócios.	334
Ao caminho da festa	335
Sua caixa de ferramentas SQL	336

Consulta externa

Consulta interna

```
SELECT alguma_coluna, outra_coluna
FROM tabela
WHERE coluna = (SELECT coluna FROM tabela);
```

*Consulta externa.*

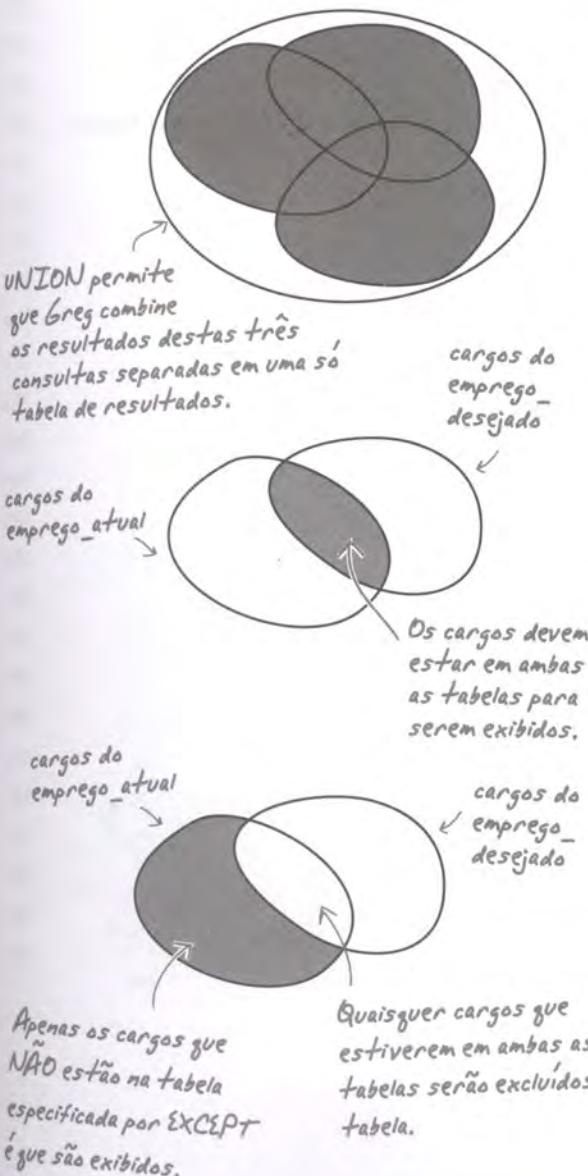
*Consulta interna.*

# 10

## Coneções externas, intraconexões e uniões

### Novas manobras

**Você só sabe metade da história sobre conexões.** Você já viu conexões cruzadas que retornam todas as linhas possíveis e consultas internas que retornam valores de ambas as tabelas onde há uma combinação. Mas o que ainda não as **consultas externas** que retornam as linhas que não possuem duplicatas compatíveis na outra tabela, **intraconexão** que (por mais estranho que pareça) conecta uma só tabela a ela mesma, e **uniões** que combinam resultados de consultas. Uma vez que tenha aprendido esses truques, você será capaz de acessar todos os seus dados exatamente da forma que precisar (e nós não esquecemos sobre expor a verdade sobre subconsultas!)



308	Limpando os dados antigos	340
309	É uma questão de esquerda e direita	340
310	Aqui está uma conexão externa esquerda	341
311	Conexões externas e combinações múltiplas	346
312	Conexão externa direita	347
313	Enquanto você estava conectando externamente...	348
314	Nós poderíamos criar uma nova tabela	349
315	Como a nova tabela se encaixa	350
316	Uma chave externa auto-referenciada	350
317	Conecte uma tabela a ela mesma	351
318	Nós precisamos de uma autoconexão (SELF-JOIN)	352
319	Outra forma de obter informações multi-tabelas	353
320	Você pode utilizar uma UNION (UNIÃO)	354
321	UNION é limitado	355
322	Regras do UNION em ação	355
323	UNION ALL	356
324	Crie uma tabela a partir da sua UNION	357
325	INSTERSECT e EXCEPT	357
326	Nós já tivemos o suficiente de conexões, hora de ir para	358
327	Subconsultas e conexões comparadas	358
328	Transformando uma subconsulta em uma conexão	359
329	Uma auto relacionamento como uma subconsulta	363
330	A companhia do Greg está crescendo	363
331	Sua caixa de ferramentas SQL	365

# 11

Constraints, views e transactions:

## **Cozinhar demais pode estragar o banco de dados**

### **Seu banco de dados cresceu, e outras pessoas precisam utilizá-lo.**

O problema é que alguns deles não serão tão hábeis como você é com SQL. Você precisa de algumas maneiras de **impedi-los de acessar dados errados**, técnicas para permiti-los **visualizar somente parte dos dados**, e maneiras de **pisarem um nos outros ao tentarem acessar os mesmos dados ao mesmo tempo**. Neste capítulo, começamos a proteger nossos dados do erro de outros. Bem-vindo aos Bancos de dados defensivos, Parte 1.

Greg contratou ajuda	370
Primeiro dia de Jim: Inserindo um novo cliente	371
Jim evita NULL	372
Antecipando três meses	372
CHECK, por favor: Adicionando uma CHECK CONSTRAINT (Restrição CHECK)	373
CHECKando o sexo	374
O trabalho de Frank ficou tedioso	375
Criando uma view	377
Visualizando suas views	377
O que sua view está realmente fazendo	378
O que é uma view	379
Inserindo, atualizando e deletando com views	382
O segredo é fingir que a view é uma tabela de verdade	382
View com CHECK OPTION	385
Sua view pode ser atualizável, se....	385
Quando você tiver terminado com sua view	386
Quando coisas ruins acontecem com bons bancos de dados	387
O que aconteceu dentro do caixa eletrônico?	388
Mais problemas no CAIXA ELETRÔNICO	389
Isto não é um sonho, é uma transação	390
O teste clássico ACID	391
SQL ajuda a gerenciar suas transações	392
O que deveria ter acontecido dentro do Caixa eletrônico	393
Como fazer as transações funcionarem com MySQL	394
Agora tente você mesmo	395
Sua caixa de ferramentas SQL	398



## Segurança

### Protegendo suas riquezas

12

**Você gastou uma enorme quantidade de tempo e energia para criar seu banco de dados.** E ficaria devastado se alguma coisa acontecesse com ela. Você também teve que dar acesso a seus dados para outras pessoas, e estava preocupado com o que eles pudessem inserir ou atualizar algo incorretamente, ou ainda pior, **deletar dados errados**. Você está prestes a aprender 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 quê com seus dados**.

370	Problema de usuários	402
371	Evitando erros no banco de dados rastreador de palhaços	402
372	Proteja a conta de usuários raiz	404
372	Adicione um novo usuário	405
373	Decida exatamente o que o usuário necessita	405
374	Um simples comando GRANT	406
375	Variações do comando GRANT	409
377	REVOKE (REVOGUE) privilégios	410
377	REVOGANDO uma GRANT OPTION usada	410
378	REVOGANDO com mais precisão	411
379	O problema com contas partilhadas	414
382	Utilizando sua role	415
382	Eliminando roles	416
385	Utilizando sua role WITH ADMIN OPTION (com opção para administrador)	417
385	Combinando CREATE USER com GRANT	421
386	A lista de Greg agora é global!	422
387	Sua caixa de ferramentas SQL	424
388	O que acha de ter uma Greg's list na sua cidade?	426
389	Utilize SQL em seus próprios projetos, e você também poderia ficar igual ao Greg	426



## sobras

**Os tópicos top 10 (que não cobrimos)**

i

**Mesmo depois de tudo isso, ainda há mais um pouco.**

Há apenas mais algumas coisas que acreditamos que você precisa saber.

Não nos sentiríamos bem em ignorá-las, ainda que elas precisam de apenas uma pequena menção. Então antes de encostar o livro, leia **estas pequenas, mas importantes migalhas**.

Além disso, quando tiver acabado aqui, tudo que falta são dois apêndices...e o índice...e talvez alguns anúncios...e então você terá realmente acabado.

**Nós prometemos!**

Nº. 1 Pegue um GUI para seu Sistema	428
Nº. 2 Palavras Reservadas e Caracteres Especiais	429
Nº. 3 ALL, ANY e SOME	431
Nº. 4 Mais Tipos de Dados	432
Nº. 5 Tabelas temporárias	434
Nº. 6 Molde seus dados	434
Nº. 7 Quem é você? Que horas são?	435
Nº. 8 Funções numéricas úteis	436
Nº. 9 Indexando para deixar as coisas mais rápidas	438
Nº. 10 PHP/MySQL em dois minutos	438

A	ABSOLUTE ACTION ADD ADMIN AFTER AGGREGATE ALIAS ALL ALLOCATE ALTER ANY ARE ARRAY AS ASC ASSERTION AT AUTHORIZATION
B	BEFORE BEGIN BINARY BIT BLOB BOOLEAN BOTH BREADTH BY
C	CALL CASCADE CASCaded CASE CAST CATALOG CHAR CHARACTER CHECK CLASS CLOB CLOSE COLLATE COLLATION COLUMN COMMIT COMPLETION CONNECT CONNECTION CONSTRAINT CONSTRAINTS CONSTRUCTOR CONTINUE CORRESPONDING CREATE CROSS CUBE CURRENT CURRENT_DATE CURRENT_PATH CURRENT_ROLE CURRENT_TIME CURRENT_TIMESTAMP CURRENT_USER CURSOR CYCLE
D	DATA DATE DAY DEALLOCATE DEC DECIMAL DECLARE DEFAULT DEFERRABLE DEFERRED DELETE DEPTH DEREF DESC DESCRIBE DESCRIPTOR DESTROY DESTRUCTOR DETERMINISTIC DICTIONARY DIAGNOSTICS DISCONNECT DISTINCT DOMAIN DOUBLE DROP DYNAMIC
E	EACH ELSE END END_EXEC EQUALS ESCAPE EVERY EXCEPT EXCEPTION EXEC EXECUTE EXTERNAL
F	FALSE FETCH FIRST FLOAT FOR FOREIGN FOUND FROM FREE FULL FUNCTION
G	GENERAL GET GLOBAL GO GOTO GRANT GROUP GROUPING
H	HAVING HOST HOUR
I	IDENTITY IGNORE IMMEDIATE IN INDICATOR INITIALIZE INITIALLY INNER INOUT INPUT INSERT INT INTEGER INTERSECT INTERVAL INTO IS ISOLATION ITERATE
J	JOIN
K	KEY
L	LANGUAGE LARGE LAST LATERAL LEADING LEFT LESS LEVEL LIKE LIMIT LOCAL LOCALTIME LOCALTIMESTAMP LOCATOR
M	MAP MATCH MINUTE MODIFIES MODIFY MODULE MONTH
N	HAMES NATIONAL NATURAL NCHAR NCLOB NEW NEXT NO NULL NOT NULL NUMERIC
O	OBJECT OF OFF OLD ON ONLY OPEN OPERATION OPTION OR ORDER ORDINALITY OUT OUTER OUTPUT
P	PAD PARAMETER PARTIAL PATH POSTFIX PRECISION PREFIX PREORDER PREPARE PRESERVE PRIMARY PRIOR PRIVILEGES PROCEDURE PUBLIC
Q	READ READS REAL RECURSIVE REF REFERENCES REFERENCING RELATIVE RESTRICT RESULT RETURN REVOKE RIGHT ROLE ROLLBACK ROLLUP ROUTINE ROW ROWS
S	SAVEPOINT SCHEMA SCROLL SCOPE SEARCH SECOND SECTION SELECT SEQUENCE SESSION SESSION_USER SET SETS SIZE SMALLINT SOME SPACE SPECIFIC SPECIFICTYPE SQL_SQLEXCEPTION SQLSTATE SQLWARNING START STATE STATEMENT STATIC STRUCTURE SYSTEM_USER
T	TABLE TEMPORARY TERMINATE THAN THEN TIME TIMESTAMP TIMEZONE_HOUR TIMEZONE_MINUTE TO TRAILING TRANSACTION TRANSLATION TREAT TRIGGER TRUE
U	UNDER UNION UNIQUE UNKNOWN UNNEST UPDATE USAGE USER USING
V	VALUES VALUES VARCHAR VARIABLE VARYING VIEW
W	WHEN WHENEVER WHERE WITH WITHOUT WORK WRITE
X	
Y	YEAR
Z	ZONE

```
> SELECT CURRENT_DATE;
+-----+
| CURRENT_DATE |
+-----+
| 2007-07-26 |
+-----+
1 row in set (0.00 sec)
```

```
> SELECT CURRENT_TIME;
+-----+
| CURRENT_TIME |
+-----+
| 11:26:48 |
+-----+
1 row in set (0.00 sec)
```

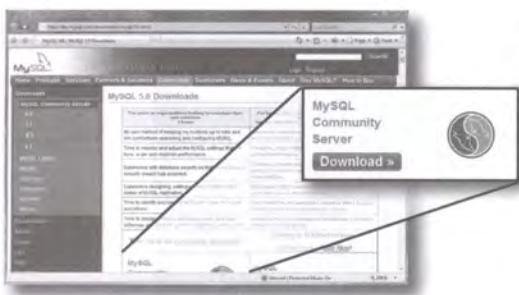
```
File Edit Window Help
SELECT CURRENT_USER;
+-----+
| CURRENT_USER |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)
```

## Instalação do MySQL

**Tente você mesmo**

**Todos as suas novas habilidades não farão muito sem um lugar para aplicá-las.** Este apêndice contém instruções para instalar seu próprio Sistema SQL para que você possa trabalhar.

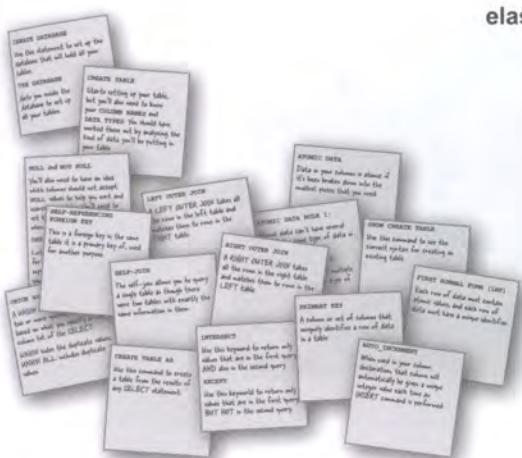
Comece logo!	442
Instruções e Solução de Problemas.	442
Passos para instalar MySQL no Windows	442
Passos para instalar MySQL em Mac OS X	445



## compilação de ferramentas

**Todas as suas novas ferramentas SQL**

Aqui todas as suas ferramentas SQL estão em um só lugar pela primeira vez, por apenas uma noite (**brincadeirinha**)! Esta é uma compilação de todas as ferramentas que abordamos. Gaste alguns minutos para analisar a lista e poder se sentir o máximo – **pois você aprendeu todas elas!!!**



Símbolos	448
A - C	448
D - E	449
G - M	450
N - S	451
T - W	452

1991  
1992  
1993  
1994  
1995

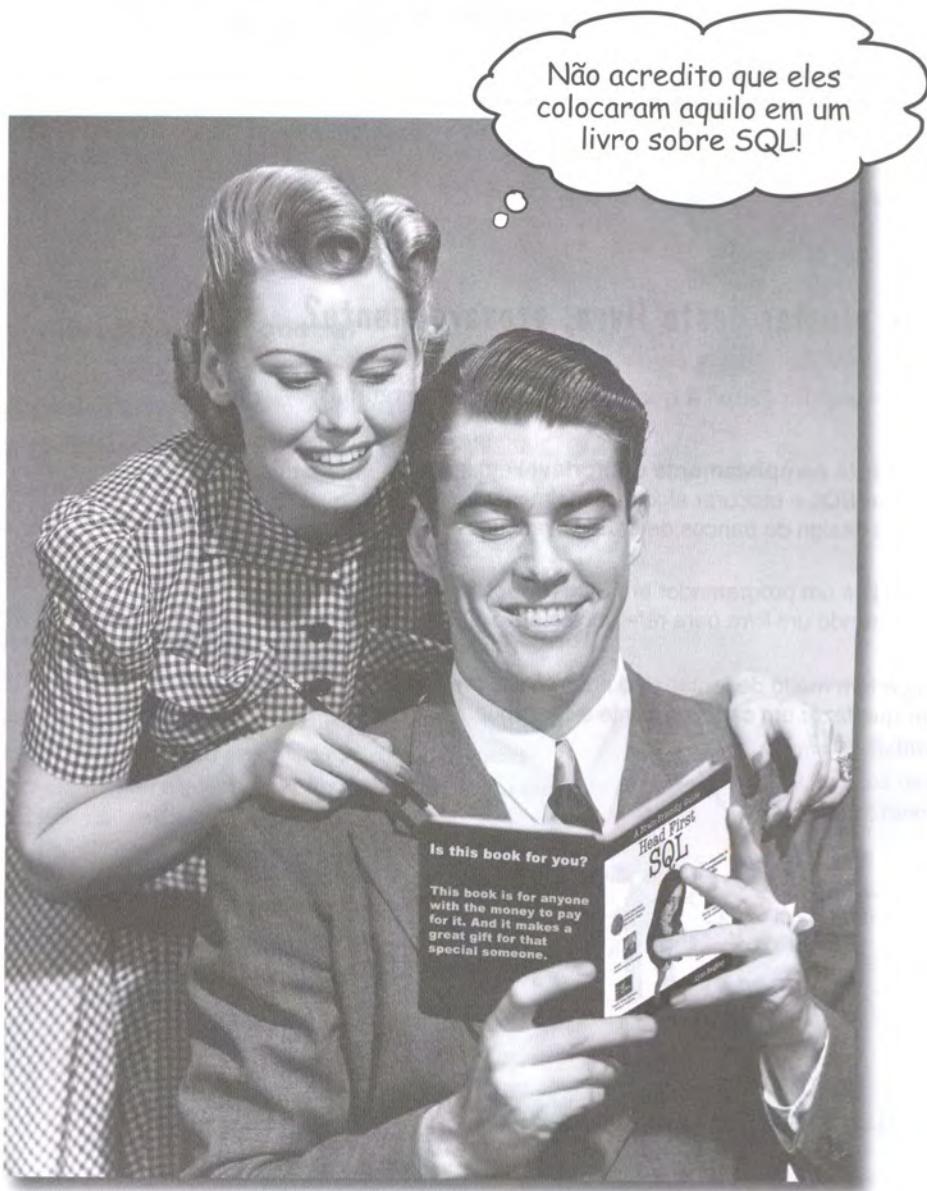


... a man of faith  
and one who sets an example  
of a good Christian life. I hope  
you will also find his message, the  
confidence he conveys in God, inspiring.



## Como usar este livro

### **Introdução**



Nesta seção, nós respondemos a uma pergunta inquietante:  
Por que eles colocaram aquilo em um livro sobre SQL?

## Para quem é este livro?

Se você puder responder “sim” para todas essas perguntas:

- ① Você tem acesso a um computador com um RDBMS instalado, como Oracle, MS SQL, ou MySQL? Ou um computador onde possa instalar o MySQL ou outro RDBMS?
- ② Você quer aprender, entender e lembrar como criar tabelas, bancos de dados, e escrever consultas usando os mais novos e modernos padrões?
- ③ Você prefere uma festa estimulante com jantar ao invés de palestras secas, desanimadas e acadêmicas?

Este livro é para você.

Nós iremos te ajudar a aprender os conceitos e sintaxe em SQL de uma maneira que definitivamente fará tudo mais fácil pra você entender e de fato usar SQL exatamente do jeito que você precisa usar.

## Quem deveria se afastar deste livro, provavelmente?

Se você responder “sim” a qualquer uma destas perguntas:

- ① Você está completamente confortável em iniciar a sintaxe SQL e procurar algo que realmente irá ajudá-lo com o design de bancos de dados avançados?
- ② Você já é um programador em SQL avançado e está procurando um livro para referências em SQL?
- ③ Você tem medo de tentar algo novo? Você prefere ter que fazer um canal no dente a usar roupa xadrez com listras?

Mas se quiser um lembrete e nunca entendeu formulários normais e conexões um-para-muitos e conexão esquerda externa, este livro pode ajudar.

Este livro não é para você.



[Observação do departamento de marketing: este livro é para qualquer um que tenha um cartão de crédito]

## Sabemos o que você está pensando

“Como esse pode ser um livro sério sobre SQL”

“Por que tanta ilustração”

“Será que posso realmente aprender deste jeito?”

## E sabemos o que seu cérebro está pensando

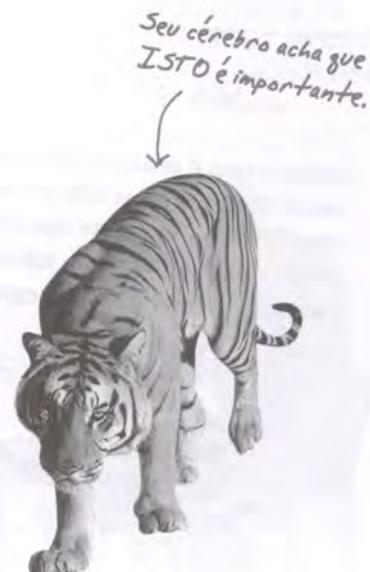
Seu cérebro anseia por novidade. Ele está sempre procurando, escaneando, esperando por algo diferente. Ele foi feito pra isto, o que ajuda a mantê-lo vivo.

Então o que seu cérebro faz com tudo que é rotineiro, comum, coisas normais com as quais você se depara? Tudo que ele pode fazer para que estas coisas não interrompam o verdadeiro trabalho do cérebro - gravar coisas que importam. Ele não se preocupa em armazenar as coisas chatas, pois elas nunca passam pelo filtro "isto obviamente não é importante".

Como o seu cérebro sabe o que é importante? Suponha que você tirou folga para fazer trilha e um tigre pula na sua frente, o que acontece dentro de sua cabeça e do seu corpo?

Os neurônios se disparam. As emoções sobem. Surge a química.

É assim que seu cérebro sabe...



### Isto deve ser importante! Não se esqueça!

Imagine, portanto, você em casa ou em uma biblioteca. É um lugar seguro, aquecido e seguro contra tigres. Você está estudando, se preparando para uma prova. Ou ainda, tentando aprender algum assunto técnico bem maçante que seu chefe exigiu que você estivesse bem afiado em uma semana ou dez dias, no máximo.

Só um problema. Seu cérebro está tentando fazer um grande favor. Ele está tentando fazer com que este assunto obviamente nada interessante não ocupe recursos escassos. Recursos que são gastos armazenando grandes coisas. Como tigres, ou o perigo que o fogo pode oferecer, por exemplo, como você nunca mais poderá praticar snowboarding de shorts.

E não há nenhuma forma simples de dizer ao seu cérebro, "Ei, cérebro, muito obrigado, mas não importa o quanto monótono este livro é, e o quanto estou registrando na escala Richter emocional neste momento, realmente quero que você mantenha essas informações acessíveis".



Nós vemos o leitor de "Use a cabeça" como um aprendiz.

Então o que é preciso para se aprender alguma coisa? Primeiro, você capta e depois se certifique de que não irá esquecer. Não diz respeito a forçar fatos para dentro de sua cabeça. Com base nas últimas pesquisas sobre ciência cognitiva, neurobiologia e psicologia educacional, aprender demanda muito mais que textos em uma página. Nós sabemos o que liga o seu cérebro.



### Algumas das principais técnicas de aprendizado de Use a cabeça:

**Faça algo ilustrado.** Imagens são de longe mais memoráveis que palavras apenas e faz o aprendizado muito mais eficiente (até 89% de progresso em estudos de recordação e transferência). Faz também com que as coisas sejam mais compreensíveis. **Coloque as palavras dentro ou perto de gráficos** que se relacionam ao texto ao invés de ao final ou até em outra página, e os aprendizes estarão duas vezes mais aptos a resolver problemas relacionados aquele conteúdo.

Bom, como sou engrapado?  
Quero dizer, engrapado  
como um palhaço, eu te  
entreterei!

### Use um estilo de conversa pessoal.

Em estudos recentes, estudantes aumentaram sua performance em 40% nos testes pós-aprendizado se o conteúdo fosse expresso diretamente ao leitor, usando a primeira pessoa, em estilo de diálogo, ao invés de optar por um tom formal. Conte histórias ao invés de palestrar. Use linguagem casual. Não leve você mesmo muito a sério. Em qual você prestaria mais atenção: um jantar com divertido com uma companhia ou uma palestra?



**Leve o aprendiz a pensar mais profundamente.** Em outras palavras, a não ser que você flexione ativamente seus neurônios, não acontece muita coisa dentro de sua cabeça. Um leitor deve estar motivado, participante, curioso e inspirado a resolver problemas, chegar a conclusões e gerar novo conhecimento. E, para isso, você precisa de desafios, exercícios e questões intrigantes e atividades que envolvam ambos os lados do cérebro e múltiplas sensações.



Espera um segundo... "Tente estas consultas", você disse. Fez parecer que todas iriam funcionar. E eu acreditei! Mas uma delas não funciona. E algumas delas não parecem que vão funcionar.

**Pegue-e-mantenha a atenção do leitor.** Nós todos já tivemos a experiência: "eu realmente preciso aprender isto, mas não consigo ficar acordado até o final da página um". Seu cérebro presta atenção a coisas que são fora do comum, interessantes, estranhas, cativantes e inesperadas. Aprender um novo, técnico e extenso assunto não precisa ser chato. Seu cérebro vai aprender muito mais rapidamente se não o for.



**Toque as emoções deles.** Agora já sabemos que sua habilidade para lembrar algo é amplamente dependente de seu conteúdo emocional. Você se lembra daquilo com o que sem importa. Você lembra quando sente algo. Não, não estamos falando de estórias melosas e emotivas sobre um garoto e seu cachorro. Estamos falando de emoções tais quais a surpresa, curiosidade, diversão, espanto! e o sentimento de que "Eu sou cara!" que vem ao resolver um quebra-cabeça, aprender algo que todo mundo pensa que é difícil, ou perceber que sabe algo que "eu sou mais técnico que você" e o Bob, da Engenharia, não sabe.

## Metacognição: Pensando sobre o pensar

Se realmente quer aprender, e mais rápido e profundamente, preste atenção em como você presta atenção. Pense sobre como você pensa. Aprenda como você aprende.

A maioria da gente nunca teve aula de metacognição ou teoria do aprendizado, enquanto crescímos. Éramos cobrados a aprender, mas muito raramente ensinados a aprender.

Mas presumimos que se você está segurando este livro, você realmente quer aprender sobre SQL e provavelmente não quer que leve muito tempo. E já que vai fazer um teste sobre isto, precisa se lembrar do que lê. E para isso você tem que entender. Para extrair o máximo desse livro, ou qualquer outro livro ou de uma experiência de aprendizado, assuma a responsabilidade para seu cérebro. Seu cérebro neste conteúdo.

O truque está em fazer com que seu cérebro enxergue um novo material que está aprendendo como sendo realmente muito importante. Crucial para sua existência. Tão importante quanto o tigre. Ou de outra forma, estará em uma batalha constante, com seu cérebro que faz o possível para evitar que um novo seja armazenado.

### **Então o que devo fazer para que meu cérebro pense que SQL é um tigre faminto?**

Há o método lento e tedioso, ou o método mais rápido e mais eficaz. O jeito lento é repetição. Você obviamente sabe que é capaz de aprender e relembrar até do mais chato assunto se ficar constantemente jogando estas informações para seu cérebro. Com a quantidade suficiente de repetição, seu cérebro diz: "Isto não me parece importante, mas ele fica pensando nesta mesma coisa várias e várias vezes, então acho que deve ser".

O modo rápido é **fazer qualquer coisa que aumente a atividade cerebral**, especialmente tipos diferentes de atividade. As coisas da página anterior são boa parte da solução e elas todas são soluções comprovadas a ajudar seu cérebro a trabalhar em seu favor. Por exemplo, estudos mostram que colocar as palavras dentro de figuras que elas expressam (ao contrário de qualquer outro lugar da página, como no topo, ou dentro de um parágrafo) causa seu cérebro a tentar fazer sentido entre a palavra e a ilustração com ela relacionada, e nesses casos mais neurônios irão disparar, mais neurônios disparando = mais chances de seu cérebro entender que isso é algo que vale a pena prestar atenção e possivelmente gravá-lo.

Um estilo de conversa ajuda porque as pessoas tendem a prestar mais atenção quando elas percebem que estão em um diálogo, onde se espera do outro o acompanhamento e a espera até o final do assunto. Uma coisa maravilhosa é, seu cérebro não precisa necessariamente se importar que a conversa seja entre você e um livro! Por outro lado, se o estilo de escrita for formal e seco, seu cérebro captará do mesmo modo que você o faria em uma palestra chata, sentado junto a uma porção de observadores passivos. Não há nem como ficar acordado.

Mas figuras e estilo de conversação são apenas o começo.



## Aqui está o que fizemos:

Nós usamos ilustrações porque seu cérebro tende para o visual e não o textual. Até onde seu cérebro sabe, uma figura vale mais que mil palavras. E quando texto e figura trabalham juntos, fixamos o texto em figuras porque seu cérebro trabalha mais eficientemente quando o texto está dentro de algo ao qual este texto se refere, ao contrário de colocá-la no topo ou enterrado em um parágrafo qualquer.

Usamos a **redundância**, dizendo a mesma coisa de diferentes formas e com tipos diferentes tipos de mídias e com múltiplos sentidos para aumentar a chance de que o conteúdo se agregue a mais de uma área de seu cérebro.

Usamos conceitos e figuras de formas **inesperadas** porque seu cérebro está afinado a novidades, e nós usamos figuras e idéias com pelo menos um pouco de conteúdo **emocional**, porque seu cérebro está afinado para prestar atenção à bioquímica das emoções. É o que provoca você a sentir algo mais provável a ser lembrado, ainda que esta emoção seja nada mais que um pouco de **humor, surpresa e interesse**.

Usamos um estilo de **conversa pessoal** porque seu cérebro está afinado a aprender e lembrar mais quando você **faz** coisas do que quando lê sobre coisas. E fizemos exercícios desafiadores, porém possíveis de serem feitos, porque é o que a maioria das pessoas prefere.

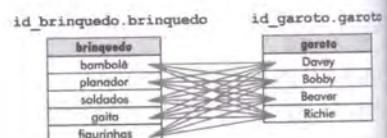
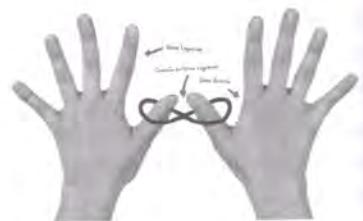
Usamos **múltiplos estilos de aprendizado**, porque você pode preferir um procedimento passo-a-passo enquanto outra pessoa prefere entender de forma generalizado primeiro, e já outra pessoa quer apenas ver um exemplo. Mas, a despeito do seu estilo preferido de aprendizado, todos serão beneficiados em ler o mesmo conteúdo representado de várias formas.

Incluímos conteúdo para **ambos os lados de seu cérebro** porque quanto mais você envolve seu cérebro, mais provavelmente aprenderá e lembrará, e mais tempo conseguirá ficar concentrado. Ainda que trabalhar apenas um lado do cérebro é a oportunidade de deixar o outro lado descansar, você terá um aprendizado muito mais produtivo e por um maior período.

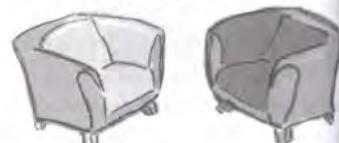
E incluímos **estórias** e exercícios que apresentam **mais de um ponto de vista**, porque seu cérebro está afinado para aprender mais profundamente quando forçado a fazer análises e julgamentos.

Incluímos desafios através de exercícios e fizemos perguntas que nem sempre têm uma resposta direta, porque seu cérebro está afinado a aprender e lembrar quando está ocupado com algo. Pense nisto – você não consegue ficar em forma apenas assistindo pessoas fazendo exercícios, mas fizemos o melhor possível para termos certeza de que quando você está trabalhando arduamente, está fazendo as coisas certas, ou seja, **não está gastando um dendrito extra** apenas pensando em exemplos incompreensíveis ou extremamente difíceis, ou com jargões técnicos, ou ainda, muito prolixos.

Usamos **pessoas** nas estórias, nos exemplos, ilustrações, etc., porque, bem, porque você é uma pessoa, e seu cérebro presta mais atenção em pessoas do que em coisas.



### Bate-papo





*Corte os tópicos e cole na sua geladeira.*

## Curve seu cérebro à submissão

Então, fizemos nossa parte, o resto é com você. Estas dicas são um ponto de partida; escute o que seu cérebro e descubra o que funciona e o que não. Tente coisas novas.

### 1 Vá devagar, quanto mais você aprende, menos tem que decorar.

Não leia apenas, pare e pense. Quando o livro faz uma pergunta, não vá direto para a resposta. Imagine que alguém realmente está fazendo uma pergunta. Quanto mais profundamente você forçar seu cérebro para pensar, melhor a chance de aprender e relembrar.

### 2 Faça os exercícios, escreva suas próprias anotações.

Nós colocamos à disposição, mas se o fizermos, será como ter alguém para malhar por você na academia. E não fique só olhando para os exercícios. **Use um lápis.** Há uma porção de evidências que a atividade física pode aumentar o aprendizado enquanto se aprende.

### 3 Leia o quadro “Não existem perguntas idiotas”

Isto quer dizer todas elas. Elas não são barras laterais opcionais – **elas são parte do conteúdo central!** Não pule!

### 4 Faça com que este seja a última coisa que você leia antes de ir para cama. Ou ao menos, a última coisa desafiadora.

Parte do aprendizado (especialmente a transferência para a memória de longa-duração) acontece depois que você encosta seu livro. Seu cérebro precisa de um tempo só dele para trabalhar mais. Se inserir algo novo nesse período, parte do que você já havia estudado irá se perder.

### 5 Beba bastante água.

Seu cérebro trabalhar ao máximo mergulhado em uma boa quantidade de fluido. Desidratação (que pode ocorrer antes de você sentir sede) diminui a função cognitiva.

### 6 Fale sobre isso. Em voz alta.

A fala ativa uma parte diferente do cérebro. Se você está tentando entender algo ou aumentar suas chances de lembrar algo mais tarde, fale em voz alta. Melhor ainda, tente explicar em voz alta para uma outra pessoa. Você aprenderá mais rapidamente e pode ainda revelar idéias que não sabia que estavam lá quando iniciou a leitura sobre o assunto.

### 7 Escute o seu cérebro.

Preste atenção em quando o seu cérebro estiver ficando sobrecarregado. Se perceber que seu cérebro está começando a pairar pela superfície do livro ou esquecer o que acabou de ler, é hora de uma pausa. Quando você atinge certo ponto, não vai aprender mais só porque está tentando soterrar mais informações, e pode ainda, afetar o processo.

### 8 Sinta alguma coisa!

Seu cérebro precisa saber que isto importa. Envolve-se com as estórias. Invente sua própria legenda para as fotos. Suspirar por causa de uma piada ruim é melhor que não sentir nada, de modo algum.

### 9 Crie alguma coisa!

Aplique isto para seu trabalho diário; Use o que você está aprendendo para tomar decisões nos seus projetos. Apenas faça alguma coisa para obter alguma experiência além dos exercícios e atividades deste livro. Tudo que você precisa é de um lápis e um problema a ser solucionado...um problema que pode até ser resolvido ao usar ferramentas e técnicas que está usando para o exame.

## Leia-me

Esta é uma experiência de aprendizado, não um livro de referência. Nós arrancamos tudo o que poderia ser uma barreira de aprender aquilo em que estamos trabalhando naquele ponto do livro. E pela primeira vez assimilado, você precisa começar do início, porque o livro supõe que você já viu ou aprendeu os assuntos anteriores.

### **Iniciamos com as sintaxes básicas SQL, então conceitos de design de bancos de dados, e então consultas avançadas.**

Enquanto é importante criar tabelas bem desenvolvidas, antes que você possa, é preciso entender a sintaxe de SQL. Então iniciam com comandos SQL que você pode tentar por si próprio. Assim, pode imediatamente fazer algo com SQL, e começará a se empolgar sobre isso. Então, um pouco depois no livro, nós mostraremos boas práticas em desenvolvimento de tabelas. Ai então terá uma compreensão sólida das sintaxes que você precisa saber, e poderá focalizar em aprender os conceitos.

### **Nós não cobrimos cada comando, função ou palavra-chave em SQL.**

Enquanto poderíamos ter colocado cada comando, função ou palavra-chave SQL em particular neste livro, pensamos que você preferiria ter um livro possível de ser carregado e que ensinaria os comandos, funções e palavras-chave mais importantes. Nós oferecemos aqueles que você precisa saber, aqueles que você usará em 95 por cento do tempo. E quando tiver lido todo o livro, terá a confiança de procurar aquela função que precisará para terminar aquela consulta que acabou de escrever.

### **Não nos referimos à espécie de Sistema de gestão de bases de dados relacionais (RDBMS).**

Existe Standard SQL, MySQL, Oracle, MS SQL Server, PostgreSQL, DB2 e mais uma porção de Sistemas de gestão de bases de dados relacionais por aí. Se cobrissemos cada variação de sintaxe para cada comando neste livro, ele teria muito mais páginas. Nós gostamos de árvores, então estamos focando em Standard SQL com um tom tendente para MySQL. Todos os exemplos no livro irão funcionar com MySQL. E a maioria funcionará em qualquer dos Sistemas de gestão de bases de dados relacionais (RDBMS) listados acima. Ainda lembra daquele livro de referência que sugerimos que você comprasse? Compre um para o Sistema SQL específico que você usa, da alta books preferencialmente.

### **As atividades não são opcionais.**

Os exercícios e atividades não são adicionais; eles são parte do conteúdo central deste livro. Alguns são para ajudar com a memória, alguns para entendimento e alguns ajudarão você a aplicar o que aprendeu. Não pule os exercícios. As palavras cruzadas são as únicas coisas que você não precisa fazer, mas são ótimas em dar ao seu cérebro uma chance de pensar nas palavras e termos que aprendido em um contexto diferente.

### **A redundância é intencional e importante.**

Uma diferença significante no livro Use a Cabeça é que nós queremos que você realmente entendam. E queremos que termine o livro lembrando-se do que aprendeu. A maioria dos livros de referência não tem o objetivo de firmar conhecimento ou ainda relembrá-lo, mas este livro é sobre aprendizado, então você verá alguns mesmos conceitos aparecerem mais de uma vez.

### **Os exemplos são os mais enxutos possíveis.**

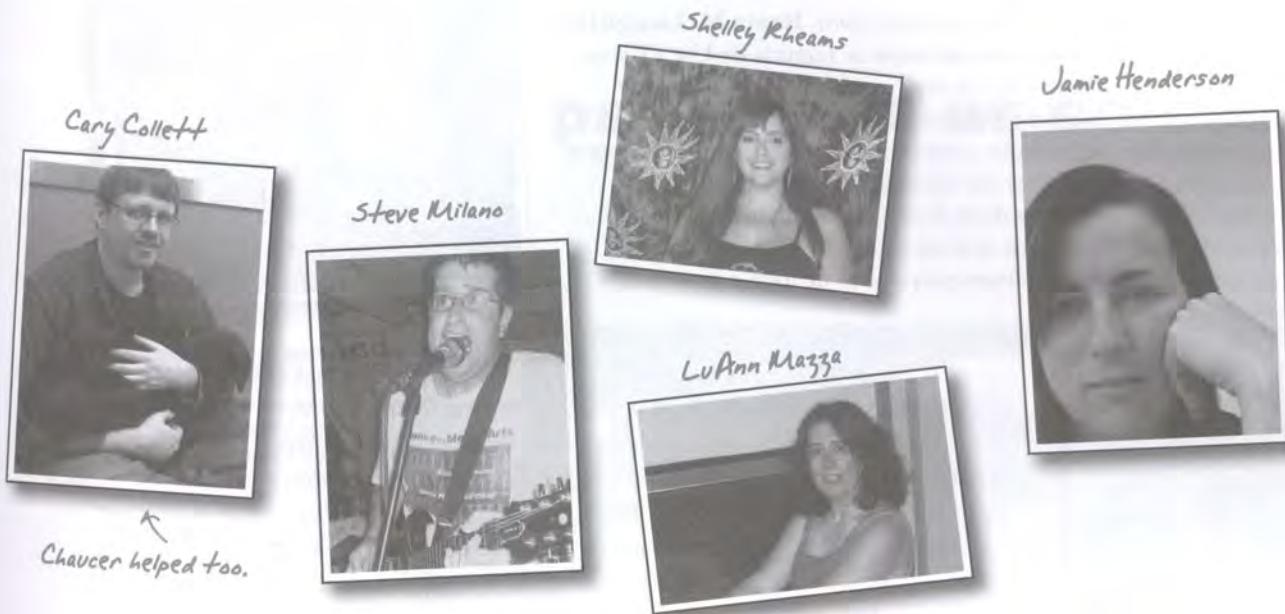
Nossos leitores nos contam que é frustrante se arrastarem em 200 linhas de exemplos procurando pelas duas linhas que eles precisam entender. A maioria dos exemplos neste livro é mostrada dentro dos mais diminutos contextos possíveis, para que a pessoa que você está tentando aprender fique clara e simples. Não espere que todos os exemplos sejam robustos ou mesmo completos - eles são escritos especialmente para aprendizado e nem sempre são integralmente funcionais.

Nós inserimos muitos dos comandos na Web para que você possa fazer download no seu computador e criar vários banco de dados utilizados neste capítulo. Você os encontrará em <http://www.altabooks.com.br>

### **Os exercícios “Poder do cérebro” não têm respostas.**

Para alguns deles, não há uma resposta correta, e para outros, parte da experiência de aprendizado das atividades do Poder do Cérebro é para que você decida se e quando suas respostas estão corretas. Em alguns dos exercícios do Poder do cérebro, você encontrará pistas para direcioná-lo ao local correto.

## O Time de Colaboração Técnica



### Nossos maravilhosos colaboradores:

Enormes agradecimentos para nosso Time de Colaboradores Técnicos. Eles capturaram erros grosseiros e sutis, e erros de digitação patéticos. Sem eles, este livro não estaria tão perto do correto como está. Eles fizeram um serviço completo em eliminar os erros deste livro.

**Cary Collett** colocou à disposição seus 15 anos de experiência trabalhando em desenvolvimento, laboratórios governamentais e, atualmente, no setor financeiro para usá-los ao revisar este livro, e está ansioso em voltar a fazer suas atividades não-empregatícias como cozinhar, fazer trilha e aterrorizar seus cachorros.

**LuAnn Mazza** encontrou tempo em sua vida profissional em Illinois como programadora e analista de sistemas para fazer algumas revisões incrivelmente detalhadas e rápidas. Nós estamos felizes porque ela agora pode gastar seu tempo livre aproveitando seus hobbies incluindo ciclismo, fotografia, computadores, música e tênis.

Quando **Steve Milano** não está programando em meia dúzia de diferentes linguagens no seu trabalho diário, fazendo um excelente trabalho de revisão de Use a Cabeça SQL, ou tocando rock punk com sua banda Onion Flavored Rings em porões sem ventilação ao redor da terra,

ele também pode ser encontrado em casa com seus gatos Ralph e Squeak.

**“Shelley” Moira Michelle Rheams**, MEd (Mestre em Educação), MCP (Profissional Certificado pela Microsoft), MCSE (Engenheiro de Sistemas Certificado pela Microsoft), leciona e gerencia o Programa de Educação de Infância precoce na Faculdade Comunitária de Delgado, em New Orleans, Campus West Bank. Atualmente ela ama disponibilizar cursos online para irem ao encontro das mudanças na comunidade de New Orleans pós-Katrina, e nós a agradecemos por estar disponível e nos encaixar em sua agenda super lotada.

**Jamie Henderson** é uma veterana arquiteta de sistemas que ostenta seu cabelo roxo e divide o tempo livre entre seu violoncelo, leitura, vídeo games e assistir a filmes em DVD.

Este time fantástico é a razão pela qual os exercícios deste livro farão o que de fato são seus propósitos e o porquê, e quando você terminar a leitura deste livro, será um programador em SQL confiante. A atenção deles aos detalhes nos mantém distantes de sermos graciosos demais, mandões demais ou, às vezes, muito esquisitões.

## Agradecimentos

### Meus editores:

Primeiramente, eu quero agradecer ao meu editor, **Brett McLaughlin**, não só por um, mas por dois acampamentos de recrutas de Use a Cabeça. Brett foi mais que um editor - ele foi uma combinação de ouvinte de minhas idéias e xerpa. Não há absolutamente nenhum jeito de que este livro fosse escrito sem sua direção, apoio e interesse. Ele não só me pegou desde a primeira entrevista, mas sua admiração pelo meu, por muitas vezes humor exagerado que fez deste livro a melhor experiência como escritora que já pude ter. Ele me deu um monte de conselhos, dicas e mais do que um pouco de treinamento através de todo o processo. Obrigado, Brett!



Brett McLaughlin



Catherine Nolan

Editora **Catherine Nolan** tem uma enorme úlcera agora, graças a algumas das inacreditáveis ondas de má-sorte que tive perto do final do processo editorial. Ela é a razão pela qual este livro não saiu em 2007, e talvez a razão pela qual ele foi publicado de fato. Foi meio como malabarismo com filhotes no final, ela não deixou cair nenhum. Eu precisei muito de uma agenda, e Catherine é a melhor organizadora de horário que já pude encontrar. E penso que fui seu maior desafio até então. Vamos torcer para que seu próximo projeto corra tudo mais tranquilo, ela merece mais do que ninguém.



Lou Barr

### O time O'Reilly:

Editora de Design **Louise Barr** tem sido tanto uma grande amiga e uma designer gráfica magnífica. De algum jeito ela foi capaz de canalizar minhas idéias malucas em uma arte impressionante que fez os conceitos difíceis parecerem muito claros. Todo este grande design é dela, e não tenho dúvida que em muitos momentos neste livro, você também vai querer agradecer a ela também.

Mas teríamos publicado com uma gama de erros se não fosse pelo time de revisão técnica e **Sanders Kleinfeld** que fez um trabalho excelente como editor de produção, ao preparar este livro para edição. Ele foi mais além, muito mais além que o chamado do dever, ele observou alguns abismos conceituais que realmente precisavam ser ligados. Obrigado, Sanders!

Finalmente, quero agradecer a **Kathy Sierra e Bert Bates** por criarem esta maravilhosa série e pelos melhores, mentais e desafiadores treinamentos que tive no meu primeiro treinamento de recrutas pra Use a Cabeça. Sem aqueles três dias, bem, eu não quero nem pensar sobre o quão difícil teria sido sem eles. E os comentários editoriais finais foram dolorosamente apurados, e amplamente melhorado neste livro.

## 1 Dados e tabelas



# Um lugar para todas as coisas

Eu costumava manter uma lista de todos os meus pacientes no papel, mas eu as esquecia constantemente! Finalmente aprendi SQL e agora não perco uma alma. Aprender sobre tabelas não vai doer nada!



**Você simplesmente não odeia esquecer as coisas?** Seja a chave do seu carro, ou aquele cupom de 25% de desconto da sua loja de departamentos preferida, ou os dados de seus aplicativos, não há nada pior do que não poder **guardar aquilo que precisa...** e quando precisa. E quando o assunto é seus aplicativos, não há lugar melhor para armazenar informações importantes do que uma **tabela**. Então vire a página, entre, e viaje pelo mundo dos bancos de **dados relacionais**.

## Definindo seus dados

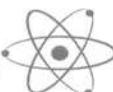
Greg conhece muitas pessoas solteiras. Ele gosta de estar por dentro sobre o que seus amigos andam fazendo, e apresentá-los uns aos outros. Ele tem várias informações sobre seus amigos em anotações garranchosas naquelas anotações adesivas como esta:



Greg tem usado este sistema por bastante tempo. Semana passada ele expandiu seus contatos, incluindo pessoas que estão procurando por emprego, assim sua lista cresceu rapidamente. Muito rapidamente....



Apenas algumas das  
anotações de Greg



## O PODER DO CÉREBRO

Há uma maneira melhor de organizar esta informação?  
O que você faria?



Bem, e o banco de dados?  
Afinal, este livro é  
sobre isso certo?

**Correto. Um banco de dados é tudo que precisamos.**

Mas antes de sair criando bancos de dados, você precisará ter uma idéia melhor de quais tipos de dados irá querer armazenar e algumas maneiras de *categorizá-los*.



## Aponte seu lápis

Aqui estão algumas das anotações de Greg. Procure por informações similares que Greg coletou sobre cada pessoa. Dê a cada pedaço de dados um nome que descreva a categoria de informação que ela é, então anote estes rótulos nos espaços abaixo.

Ann Branson  
Aniversário: 01/07/1962  
Engenheiro de Software  
Solteiro, mas comprometido.  
Mountain View, CA  
annie@boards-r-us.com  
Interesses: Colecionar  
livros, fabricação de cerveja,  
equitação.  
Procura: Um novo emprego

Jamie Hamilton  
Aniversário: 10/09/1964  
Analista de Sistemas  
Solteiro  
Sunnyvale, CA  
dontbother@breakneckpizza.com  
Interesses: Trilha, escrever.  
Procura: Amigos, Mulheres  
para relacionamento

## Procura

Angelina Mendoza  
Aniversário: 19/08/1979  
Casada  
San Francisco, CA  
Angelina@starbuzzcoffee.com  
Interesses: Atuar, dançar.  
Procura: Novo emprego

Alan Soukup  
Aniversário: 01/07/1966  
Engenheiro aeronáutico  
Casado  
San Antonio, TX  
soukup@breakneckpizza.com  
Interesses: RPG, programação.  
Procura: Nada



## Aponte seu lápis

### Solução

Aqui estão algumas das anotações de Greg. Procure por informações similares que Greg coletou sobre cada pessoa. Dê a cada pedaço de dados um nome que descreva a categoria de informação que ela é, então anote estes rótulos nos espaços abaixo.

Primeiro nome

Ann Branson

Aniversário: 01/07/1962

Engenheiro de Software

Solteiro, mas comprometido.

Mountain View, CA

annie@boards-r-us.com

Interesses: Colecionar livros, fabricação de cerveja, equitação.

Procura: Um novo emprego

Estado civil

Solteiro, mas comprometido.

Nós dividimos o nome em nome e sobrenome. Isto vai te ajudar a separar os dados mais tarde.

Agora que já criamos nossas categorias, podemos organizar nossos dados.

Sobrenome

Jamie Hamilton

Aniversário: 10/09/1964

Analista de Sistemas

Solteiro

Sunnyvale, CA

dontbother@breakneckpizza.com

Interesses: Trilha, escrever.

Procura: Amigos, Mulheres para relacionamento

Primeiro nome

Sobrenome

Aniversário

Profissão

Estado civil

Local

Email

Interesses

Procura

Profissão

Local

Email

Alan Soukup

Aniversário: 01/07/1966

Engenheiro aeronáutico

Casado

San Antonio, TX

soukup@breakneckpizza.com

Interesses: RPG, programação.

Procura: Nada

Angelina Mendoza

Aniversário: 19/08/1979

Casada

San Francisco, CA

Angelina@starbuzzcoffee.com

Interesses: Atuar, dançar.

Procura: Novo emprego

Greg já havia dado nome a algumas categorias, como Aniversário, interesses e Procura em suas anotações.

## Olhe para seus dados em categorias

Vamos olhar para seus dados de uma forma diferente. Se você cortar cada anotação em pedaços e espalhar os pedaços horizontalmente irá obter algo parecido com isto:



E se, então, cortar outra anotação adesiva com as categorias que identificou e colocá-las, em pedaços acima da informação correspondente, você terá algo parecido com isto:



Aqui está essa mesma informação disposta em uma **TABELA** em **linhas** e **colunas**.

Ok, Eu já vi dados dispostos desta forma no Excel. Mas uma tabela SQL é diferente? E o que você quer dizer com linhas e colunas?



sobrenome	primeiro_nome	email	aniversario	profissao	local	estado_civil	interesses	procura
Branson	Ann	annie@boards-r-us.com	01/07/1962	Engenheiro aeronáutico	San Antonio, TX	Solteiro, mas comprometido	RPG, programação	Novo Emprego
Hamilton	Jamie	dontbother@eakneckpizza.com	10/09/1964	Analista de Sistemas	Sunnyvale, CA	Solteiro	Trilha, escrever	Amigos, Mulheres para relacionamento
Soukup	Alan	soukup@breakneckpizza.com	02/12/1975	Engenheiro aeronáutico	San Antonio, TX	Casado	RPG, programação	Nada
Mendoza	Angelina	angelina@starbuzzcoffee.com	19/08/1979	Administrador Unix System	San Francisco, CA	Casada	Atuar, dançar	Novo emprego



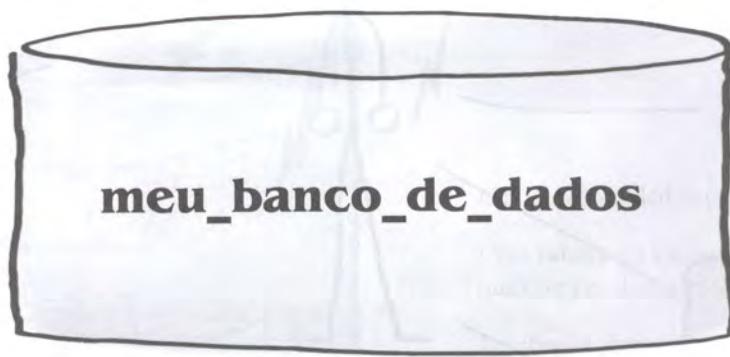
## O que fica em um banco de dados?

Volta pelo Banco de Dados

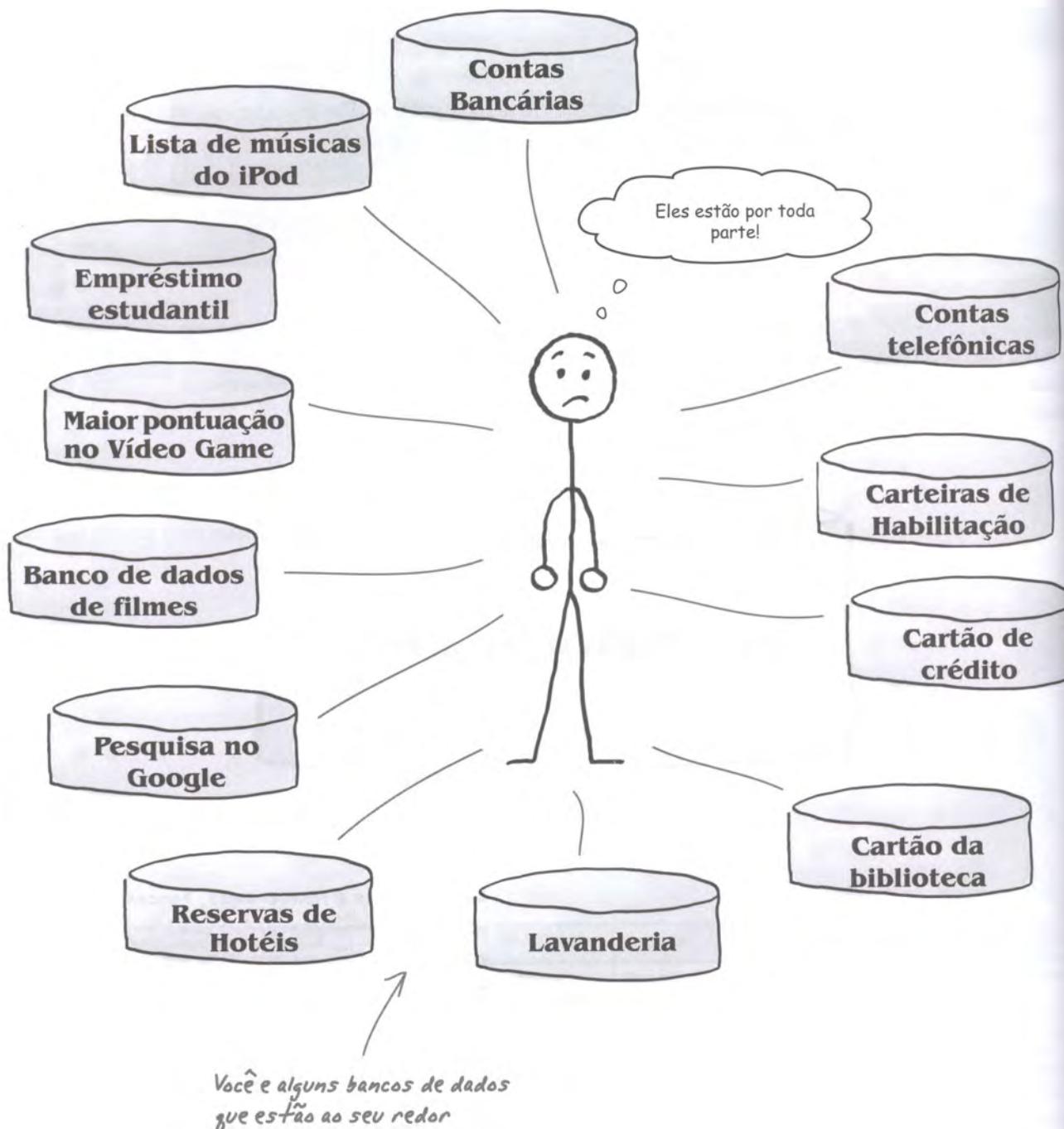
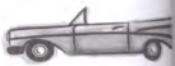
Antes de entrarmos em detalhes sobre o que são tabelas, linhas e colunas, vamos nos afastar e ter uma visão panorâmica. A primeira estrutura SQL que você precisa conhecer é o contêiner que guarda todas as tabelas, conhecida como banco de dados.

**Um banco de dados** é o contêiner que guarda todas as tabelas e outras estruturas SQL relacionadas àquelas tabelas.

Toda vez que pesquisar online, ir às compras, ligar para uma central de informações, usar seu TiVo, fazer uma reserva, ser multado por excesso de velocidade ou ir ao supermercado, um banco de dados estará sendo perguntado por informações, ou também conhecido como uma Consulta.



Em diagramas e fluxogramas, bancos de dados estão descritos como cilindros. Então, ao ver isto, pensem em banco de dados.



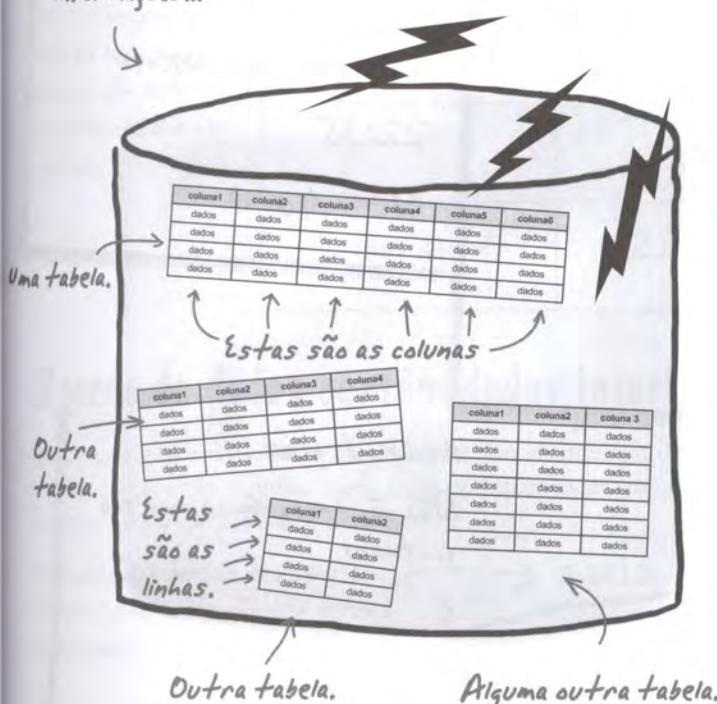


Volta pelo Banco de Dados



## Anatomia de um banco de dados

Pense no banco de dados como um contêiner que guarda informações...



Seu banco de dados visto através de uma visão raio-x...

Um banco de dados contém tabelas.

Uma **tabela** é a estrutura interna de um banco de dados que contém dados em **linhas** e **colunas**.

Lembra-se daquelas categorias que você criou? Cada categoria se torna uma **coluna** em sua tabela. Os valores deverão estar na mesma coluna: Solteiro, Casado, Divorciado.

Uma **linha** da tabela contém todas as informações sobre um objeto na tabela. Na nova tabela de Greg, uma linha seria todos os dados sobre uma pessoa. Aqui está um exemplo de alguns dados que poderiam estar na linha: John, Jackson, solteiro, escritor, jj@boards-r-us.com.

As informações dentro de um banco de dados estão organizadas em **tabelas**.



Volta pelo Banco de Dados

## Sinta-se uma tabela



Abaixo, você encontrará algumas anotações adesivas e uma tabela. Seu trabalho é ser a tabela parcialmente formada e preencher os espaços vazios para obter uma paz interior. Depois de fazer este exercício, vire a página e veja se você tornou-se uma só com a tabela.

Ducans Donuts

5

25/4

Recheio de geleia

8:56

Gorduroso

Starbuzz Coffee

23/4

Recheio de geleia

7:43

Quase perfeito

Ducans Donuts

7

25/4

Não tem geleia suficiente

22:35

Recheio de geleia

Recheio de geleia

Não são fresquinhos, mas são gostosas

6

Krispy King

26/4

21:39

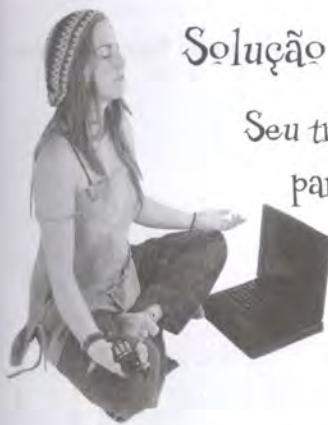
Use cada um dos campos como  
título que dará a tabela um nome  
significativo.

Empresa				
			9	
		25/04	8	
				não tem geleia suficiente



## Solução de Sinta-se uma Tabela

Volta pelo Banco de Dados



Seu trabalho era ser uma tabela  
parcialmente formada e preencher  
os espaços vazios para  
aumentar a paz interior.

Você deve estar apto a descobrir qual  
seria o título da tabela a partir das  
anotações adesivas.

Meus\_Lanches

Não se preocupe  
se suas respostas  
para os nomes das  
colunas não baterem  
exatamente com os  
nossos.

Empresa	Horario	data	nota	comentarios
Starbuzz Coffee	7:43	23/04	9	quase perfeito
Duncans Donuts	8:56	25/04	5	gorduroso
Krispy King	21:39	24/04	6	não são fresquinhos, mas são gostosos
Duncans Donuts	22:35	24/04	7	não tem geléia suficiente

## Bancos de dados contêm dados interligados

Todas as tabelas em um banco de dados devem se interligar de alguma forma. Por exemplo, aqui estão algumas tabelas que poderiam estar em um banco de dados contendo informações sobre donuts:

Os nomes dos bancos de dados e  
tabelas geralmente não têm letras  
maiúsculas.

Aqui está um banco de dados com  
três tabelas. Este banco de dados é  
chamado de meus\_lanches.

Tabela contendo informação sobre  
os donuts com geleia.

Tabela contendo informação  
sobre lanches que não são Donuts.

Empresa	Horario	data	nota	comentarios
Starbuzz Coffee	7:43 am	23/4	9	quase perfeito
Duncan's Donuts	8:56 am	25/4	5	gorduroso
Krispy King	9:39 pm	26/4	6	não são fresquinhos, mas gostosos
Duncan's Donuts	10:35 pm	24/4	7	não tem geléia suficiente

Empresa	Horario	data	nota	comentarios
Krispy King	9:39 pm	26/4	8	morno, mas não quente
Starbuzz Coffee	7:43 am	23/4	4	não tem cobertura suficiente
Duncan's Donuts	8:56 am	25/4	6	gorduroso
Duncan's Donuts	10:35 pm	24/4	7	não são fresquinhos

Empresa	Horario	data	bolo	nota	comentarios
Starbuzz Coffee	10:35 pm	24/4	bolo de canela	6	tempero em excesso
Starbuzz Coffee	7:43 am	23/4	chocolate c/ castanha	8	marshmallows!
Krispy King	9:39 pm	26/4	bala de cereais	4	poucas frutas
Duncan's Donuts	8:56 am	25/4	biscoito	9	quente, crocante

Tabela contendo informações  
sobre donuts com  
coberturas



## Visão de perto: Tabelas

Uma **coluna** é um tipo de dados armazenado em uma tabela. Uma **linha** é uma série de colunas que descrevem atributos de um único objeto (registro). Colunas e linhas juntas formam uma tabela.

Aqui está um exemplo de como uma lista de endereços contendo suas informações pessoais se parece. Você comumente verá a palavra **campo** ao invés de **coluna**. Elas querem dizer a mesma coisa. Neste mesmo sentido, **linha** e **registro** são utilizados como sinônimos.

Estas são as colunas

primeiro_nome	sobrenome	endereco	cidade	estado	Identificador
Joe	Epps	dados	dados	dados	dados
Al	Jones	dados	dados	dados	dados
Mary	Morris	dados	dados	dados	dados
Lou	Green	dados	dados	dados	dados

Estas são as linhas

Coloque as colunas e as linhas juntas e  
você terá formado uma tabela!

primeiro_nome	sobrenome	endereco	cidade	estado	identificador
Joe	Epps	dados	dados	dados	dados
Al	Jones	dados	dados	dados	dados
Mary	Morris	dados	dados	dados	dados
Lou	Green	dados	dados	dados	dados

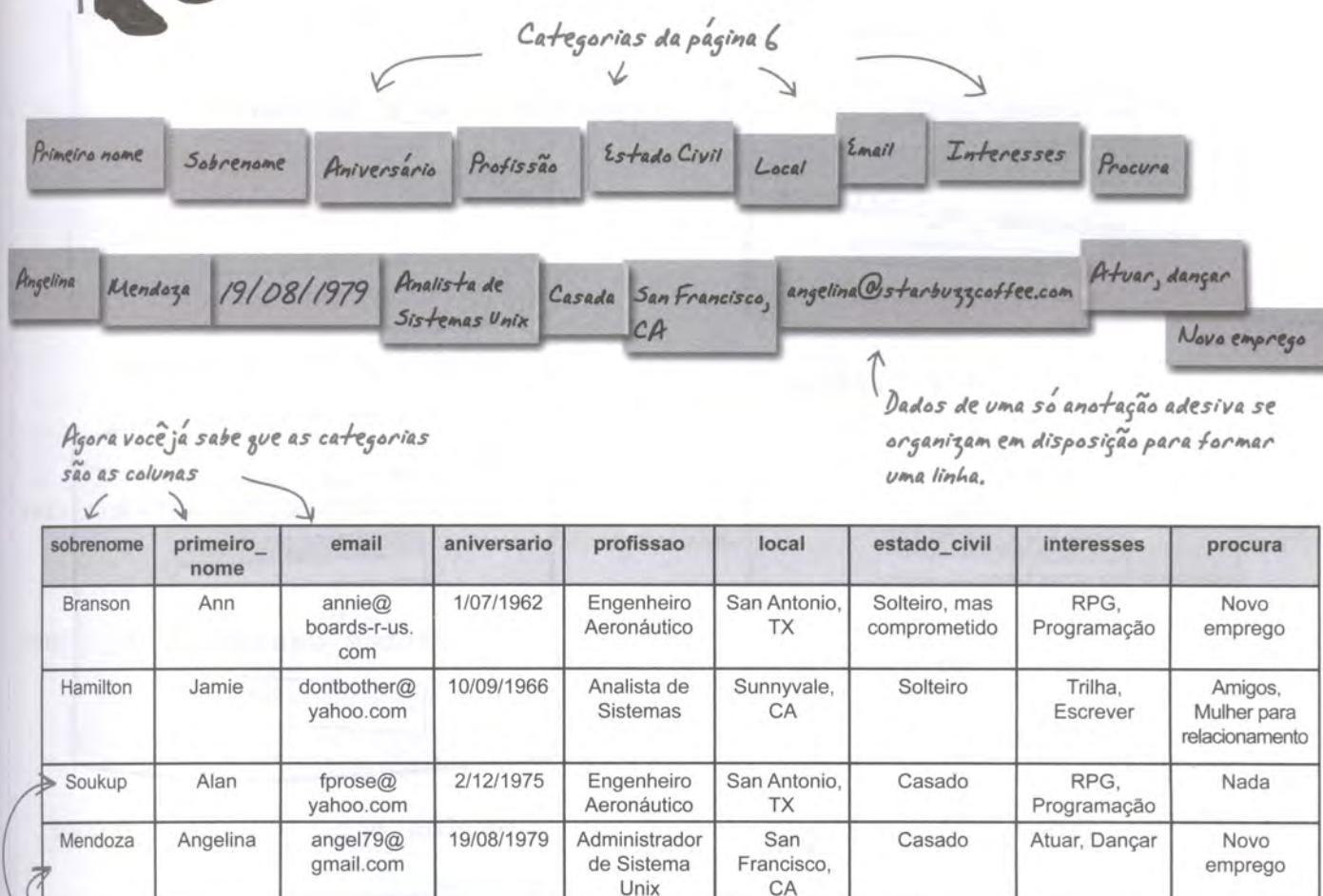
Dados



Quer dizer então que temos dados suficientes nas minhas anotações adesivas para criarmos uma tabela?

Correto! Você pode identificar as categorias para os tipos de dados que estiver agrupando para cada pessoa.

Suas categorias então se tornam as suas colunas. Cada anotação adesiva será uma linha. Você pode pegar todas as informações armazenadas naqueles recadinhos e transformá-las em uma tabela.



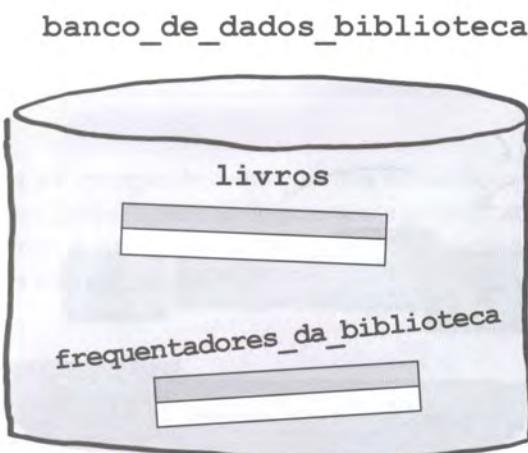
Finalmente. Ok, então como crio a minha tabela?





## Exercícios

Considere os bancos de dados e as tabelas abaixo. Pense sobre que tipos de categorias e dados você encontrará em cada uma delas. Estabeleça a provável coluna para cada tabela.



Banco de dados para uma biblioteca

livros:.....

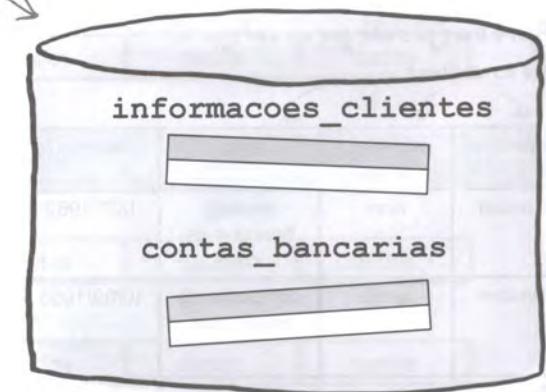
frequentadores\_da\_biblioteca:.....

Banco de dados para um banco

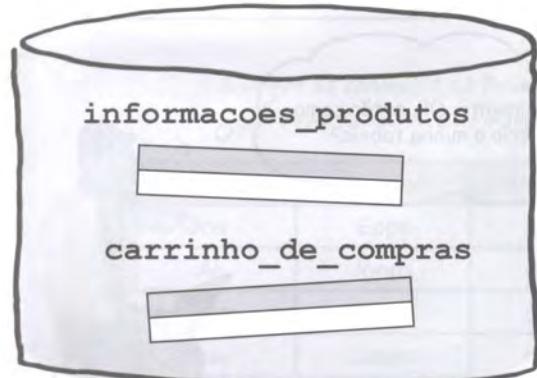
informacoes\_clientes:.....

contas\_bancarias:.....

banco\_de\_dados\_banco



banco\_de\_dados\_loja\_online



Banco de dados para uma loja online

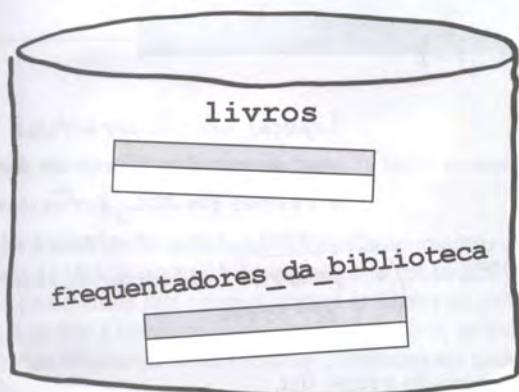
informacoes\_produtos:.....

carrinho\_de\_compras:.....



## Solução dos Exercícios

Considere os bancos de dados e as tabelas abaixo. Pense sobre que tipos de categorias e dados você encontrará em cada uma delas. Estabeleça a provável coluna para cada tabela.



## Banco de dados para uma biblioteca

banco\_de\_dados\_biblioteca

Não se preocupe se suas respostas para os nomes das colunas não baterem exatamente com as nossas respostas.

livros: título, autor, custo, código-barra

frequêntadores\_da\_biblioteca: .....  
.....

Banco de dados para um banco

contas bancarias: ..... extratos, depósitos, saques

banco de dados banco

## informações clientes

## contas\_bancarias

**ANSWER**

banco de dados loja online

## ← Banco de dados para uma loja online

The diagram shows a cylinder with two horizontal rectangular cutouts, one near the top and one near the bottom. The text "informacoes\_produtos" is written above the top cutout, and the text "carrinho\_de\_compras" is written above the bottom cutout.

informações produtos: ..... nome, tamanho, custo

carrinho de compras: *total cobrado, código cliente*

## Tome o comando!

Inicie o seu sistema de gestão de bancos de dados relacionais SQL (RDBMS) e abra uma janela de linha de comando ou ambiente gráfico que permita que você se comunique com suas RDBMS. Aqui está nossa janela do terminal, após iniciarmos o MySQL.

```
File Edit Window Help CommandMeBaby
Welcome to the SQL monitor. Commands end with ; or \g.

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

>
```

O sinal de maior é o prompt de comando. Você digitará seus comandos logo após ele.

Primeiro você precisará criar um banco de dados para armazenar todas as suas tabelas.

Espaços não são permitidos nos nomes dos bancos de dados e tabelas em SQL, então uma sublinha, como separadora de palavras (underscore), poderá ser usada.

1

Digite a linha de código abaixo para criar um banco de dados chamado `gregs_list`.

`CREATE DATABASE gregs_list;`

`CREATE DATABASE` é o comando.

O nome do banco de dados é `gregs_list`

Seu comando deve acabar com um ponto-e-vírgula.

```
File Edit Window Help CommandMeBaby
> CREATE DATABASE gregs_list;
Query OK, 1 row affected (0.01 sec)
```

Esta é a resposta do RDBMS para que saiba que sua consulta foi executada com sucesso.



**Veja Isto!**

Você leu a introdução?

Estamos usando MySQL para controlar nossos bancos de dados, portanto, os comandos no seu Sistema de gestão de Bancos de Dados (DBMS) poderá parecer um pouco diferente. Veja Apêndice II para instruções de instalação de MySQL no seu servidor.

- 2 Agora precisa dizer ao seu software de banco de dados para de fato utilizar o banco de dados que você acabou de criar.

```
File Edit Window Help USEful
> USE gregs_list;
Database changed
```

*Agora tudo que fizermos será  
em cima do banco de dados  
gregs\_list!*

---

## não existem Perguntas Idiotas

---

P: Por que preciso criar um banco de dados se tenho apenas uma tabela?

R: A linguagem SQL requer que todas as tabelas estejam inseridas em um banco de dados. Há boas razões por detrás disso. Uma das características do SQL é sua habilidade para controlar acessos às tabelas por múltiplos usuários. Estar apto a permitir ou negar acesso a todo banco de dados é, às vezes, mais simples que ter que controlar permissões em cada uma das muitas tabelas.

P: Percebi que foram usadas apenas letras maiúsculas para o comando CREATE DATABASE. Isto é necessário?

R: Alguns sistemas de fato requerem que as palavras-chave sejam em maiúsculas, mas SQL não é case sensitive (sensível ao tamanho da letra). Isto quer dizer que não é necessário colocar os comandos em maiúsculas, mas é considerado uma boa prática de programação em SQL. Observe o comando que digitamos:

```
CREATE DATABASE
gregs_list;
```

As letras maiúsculas tornam mais fácil identificar um comando (CREATE DATABASE) a partir do nome do banco de dados (gregs\_list).

P: Há alguma coisa que eu deveria saber quanto à nomenclatura dos bancos de dados, tabelas e colunas?

R: Geralmente é uma boa idéia criar nomes explicativos. Às vezes isto fará com que utilize nomes compostos. Não é possível criar nomes separados por espaços, então as sublinhas permitem que se criem nomes mais explicativos. Seguem abaixo algumas variações que você poderá ver:

```
gregs_list
gregslist
Gregslist
gregsList
```

Normalmente é melhor evitar capitalizar os seus nomes a fim de evitar confusão, pois SQL não é sensível ao tamanho da letra.

**Letras maiúsculas e  
sublinhas ajudam a  
programar em SQL (ainda  
que SQL não precise delas!)**

# Preparando a tabela: o comando CREATE TABLE

Vamos ver tudo isto nos dados sobre donuts. Digamos que esteja tendo problemas para lembrar o tipo de donuts que comeu no lanche apenas pelo seu nome, você poderá criar uma tabela para poupará-lo de lembrar deles. Abaixo há um comando único para ser digitado na janela do console. Após ter digitado, aperte ENTER para dizer ao seu SQL RDBMS para carregar o comando.

lista\_donut

nome_donut	tipo_donut
Blooberry	recheado
Cinnamondo	circular
Rockstar	enrolado
Carameller	enrolado
Appleblush	recheado

Cria

Voc  
anota  
TABL

Aqui está o comando SQL para criar as tabelas - note as letras maiúsculas.

O parêntese aberto abre a lista de colunas a serem criadas.

O nome da sua tabela deve ser minúscula e ter uma sublinha ao invés de espaços.

**CREATE TABLE lista\_donut**

Apenas aperte ENTER para iniciar uma nova linha no seu comando para ficar mais fácil para ler o que é cada coisa.

A vírgula separa as colunas a serem criadas.

O nome da primeira coluna na tabela.

**nome\_donut VARCHAR (10)**

O nome da segunda coluna.

**tipo\_donut VARCHAR (6)**

O parêntese fechado fecha a lista de colunas.

O ponto-e-vírgula diz ao SQL RMBDS que aquele é o final do comando.

) ;

Este é um DATA TYPE (tipo de dado). Ele subentende VAR de variável e CHAR de caractere (character, em inglês) e usado para guardar informações que estão armazenadas como textos. O (6) significa que a quantidade máxima que aquele campo poderá armazenar é equivalente a 6 caracteres.



Ei! E eu? E sobre um **CREATE TABLE** para o meu banco de dados gregs\_list?

## Criando uma tabela um pouco mais complicada

Você se lembra das colunas para a tabela do Greg? Nós as escrevemos numa anotação adesiva. Você precisará delas para escrever seu comando CREATE TABLE.

Você precisará do comando **CREATE TABLE** para fazer isto...

... se transformar nisto

sobrenome

primeiro nome

email

aniversário

profissão

local

estado civil

interesses

procura

sobrenome	primeiro_nome	email	aniversario	profissao	local	estado_civil	interesses	procura



**PODER DO CÉREBRO**

De que forma os nomes das colunas contidos na anotação adesiva se diferem daqueles na tabela acima? Por que isto é importante?



## Veja como é fácil escrever em SQL

Você tem visto que para criar uma tabela é necessário categorizar seus dados em colunas. Então, defina o tipo de dados e tamanho de cada coluna. Depois de estimar o tamanho que a coluna precisará ter, escrever o código é como andar em linha em reta.

CREATE



## Aponte seu lápis

O código à esquerda é o nosso comando CREATE TABLE para o novo banco de dados de Greg. Tente adivinhar o que cada linha do comando CREATE TABLE está fazendo. Inclua também um exemplo de informação que irá em cada coluna.

```
CREATE TABLE meus_contatos
(
    sobre_nome VARCHAR (30),
    primeiro_nome VARCHAR (20),
    email VARCHAR (50),
    aniversario DATE,
    profissao VARCHAR (50),
    local VARCHAR (50),
    estado_civil VARCHAR (20),
    interesses VARCHAR (100),
    procura VARCHAR(100)
);
```

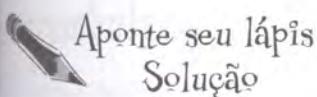
## Finaln

Agora que  
comando  
código no

Ou você p

Qualquer  
vírgula, ce

last\_na  
lastnam



Aqui está o que cada linha do comando CREATE TABLE está fazendo e alguns exemplos de dados para cada tipo de coluna.

```
CREATE TABLE meus_contatos
```

```
(  
    sobre_nome VARCHAR(30),  
    primeiro_nome VARCHAR(20),  
    email VARCHAR(50),  
    aniversario DATE,  
    profissao VARCHAR(50),  
    local VARCHAR(50),  
    estado_civil VARCHAR(20),  
    interesses VARCHAR(100),  
    procura VARCHAR(100)  
);
```

	Criar uma tabela nomeada <code>meus_contatos</code>	
	Abre a lista de colunas para ser adicionada	
sobre_nome VARCHAR(30),	Adiciona uma coluna chamada <code>'sobrenome'</code> que pode conter até 30 caracteres	<code>'Anderson'</code>
primeiro_nome VARCHAR(20),	Adiciona uma coluna chamada <code>'primeiro_nome'</code> que pode conter até 30 caracteres	<code>'Jillian'</code>
email VARCHAR(50),	Adiciona uma coluna chamada <code>'email'</code> que pode conter até 50 caracteres	<code>jill_anderson@breakneckpizza.com</code>
aniversario DATE,	Adiciona uma coluna chamada <code>'aniversario'</code> que pode conter datas	<code>'05/09/1980'</code>
profissao VARCHAR(50),	Adiciona uma coluna chamada <code>'profissao'</code> que pode conter até 50 caracteres	<code>'Escritora Técnica'</code>
local VARCHAR(50),	Adiciona uma coluna chamada <code>'local'</code> que pode conter até 50 caracteres	<code>'Palo Alto, CA'</code>
estado_civil VARCHAR(20),	Adiciona uma coluna chamada <code>'estado_civil'</code> que pode conter até 20 caracteres	<code>'Solteiro'</code>
interesses VARCHAR(100),	Adiciona uma coluna chamada <code>'interesses'</code> que pode conter até 100 caracteres	<code>'Caiaque, Répteis'</code>
procura VARCHAR(100)	Adiciona uma coluna chamada <code>'procura'</code> que pode conter até 100 caracteres	<code>'Relacionamento, Amigos'</code>
	Termina a lista de colunas para adicionar e ponto-e-vírgula finaliza o comando	

## Finalmente, criando a tabela `meus_contatos`,

Agora que sabe exatamente o que cada linha está fazendo, você pode digitar no comando **CREATE TABLE**. Você pode digitar uma linha de cada vez, copiando o código no topo desta página.

Ou você pode digitar tudo como se fosse uma grande e única linha:

```
CREATE TABLE my_contacts(last_name VARCHAR(30), first_name VARCHAR(20), email VARCHAR(50), birthday DATE, profession VARCHAR(50), location VARCHAR(50), status VARCHAR(20), interests VARCHAR(100), seeking VARCHAR(100));
```

Qualquer das maneiras que escolher, antes de apertar o enter depois do ponto-e-vírgula, certifique-se de que não deixou faltar nenhum caractere:

`last_name` VARCHAR(3) é uma coluna bem diferente de `lastname` VARCHAR(30) !

↑  
Acredite, este realmente é o código, só está escrito b-e-m pequeno para caber na página!

## Sua tabela está pronta

```
File Edit Window Help AllDone  
> CREATE TABLE meus_contatos  
-> (   
->     sobrenome VARCHAR(30) ,  
->     primeiro_nome VARCHAR(20) ,  
->     email VARCHAR(50) ,  
->     aniversario DATE ,  
->     profissao VARCHAR(50) ,  
->     local VARCHAR(50) ,  
->     estado_civil VARCHAR(20) ,  
->     interesses VARCHAR(100) ,  
->     procura VARCHAR(100)  
-> );  
Query OK, 0 rows affected (0.07 sec)
```

Você percebeu que ao apertar enter após o ponto-e-vírgula, o comando foi finalizado e disse ao seu Sistema SQL para processá-lo?

Então irei sempre gravar tudo que preciso em dados do tipo VARCHAR e DATE?



Na verdade, você precisará de alguns outros tipos de dados, como números.

Suponha que nós adicionamos uma coluna de preço para nossa tabela de donuts. Não precisaríamos armazená-la em dados tipo VARCHAR. Valores armazenados em VARCHAR são interpretados como texto, assim sendo, você não estará apto a realizar operações matemáticas neste tipo de dados. Mas há outros tipos de dados que não lhe foram apresentados ainda...



**PODER DO  
CÉREBRO**

Antes de ir mais além, pense em alguns tipos de dados que precisam ser diferentes de VARCHAR ou DATE.

## Perambulando por aí

Estes são alguns dos tipos de dados mais comuns. O trabalho deles é armazenar seus dados sem bagunçá-los. Você já conheceu VARCHAR e DATE, mas diga olá para estes.

**CHAR ou CHARACTER.** Ele é rígido e prefere que seus dados sejam de um tamanho padrão.

**INT ou INTEGER.** Ele acha que os números deviam ser sempre inteiros, mas ele não tem medo de números negativos.

**DEC, abreviação para DECIMAL.** Ele dará todas as casas decimais que você pedir, até ele se encher.

Ela é chamada tanto de DATETIME quanto TIMESTAMP, dependendo do sistema SQL RDBMS utilizado. Ela cuida de datas e horas. Ela tem um irmão gêmeo, TIME (horário), que não se importa que data é.

**VARCHAR** pode conter dados de textos de até 255 caracteres de tamanho. Ela é flexível e se adapta ao tamanho de seus dados.



Nós não sabemos quem ele é, ele estava perambulando por aí.

Chame-o de BLOB. Ele gosta de grandes blocos de dados em texto.

DATE mantém o controle de suas datas. Embora, não ligue para horários.



Estes nomes de tipos de dados podem não funcionar no seu sistema SQL RDBMS!

Infelizmente, não há uma universalidade de nomes aceitos para vários tipos de dados. Seu programa de SQL específico pode usar diferentes nomes para um ou mais tipos de dados. Cheque seus documentos e verifique quais são os nomes corretos para seu RDBMS.



## QUE TIPO DE DADOS?

Determine que tipo de dados faz mais sentido para cada coluna. E quando fizer isso, preencha as outras informações pendentes.

*Estes dois números mostram quantos dígitos o banco de dados deve esperar em frente dos decimais, e quantos depois.*

Nome da Coluna	Descrição	Exemplo	Melhor opção de Tipos de Dados
preço	O custo de um item à venda	5678.39	DEC(5,2) ←
cep			
peso_atomico	Peso atômico de um objeto com seis casas decimais		
comentarios	Bloco de texto grande, mais de 255 caracteres	Joe, estou na reunião de acionistas. Eles acabaram de fazer uma demonstração e há elásticos voando em direção à tela. Essa foi sua idéia de piada? Você precisa passar um tempo em Monster.com quantidade	
quantidade	A quantidade deste item disponível no estoque		
percentual_de_imposto		3.755	
título_do_livro		Use a Cabeça SQL	
sexo	Um caractere, M ou F		CHAR(1)
numero_do_telefone	Dez dígitos, sem pontuação	2105552367	
estado	Abreviação em dois caracteres para um estado	RJ, SP	
aniversario		22/11/2006	DATE
jogos_vencidos			INT
hora_do_encontro		10:30 12/04/2020	

---

*não existem*  
Perguntas Idiotas

---

**P:** Por que não usar apenas o BLOB para todos os tipos de dados em texto?

**R:** Seria um gasto de espaço. Uma VARCHAR ou CHAR ocupa um tamanho específico de tamanho, nada superior a 256 caracteres. Mas um BLOB ocupa muito mais espaço. À medida que seu banco de dados cresce, você corre o risco de que ele fique sem espaço no seu HD. Você também não pode executar certas operações nos BLOBS que faria no VARCHARs e CHARs (aprenderá mais sobre isso posteriormente).

**P:** Por que preciso dos tipos numéricos como INT e DEC?

**R:** Tudo se resume em tamanho e eficiência de armazenamento de um banco de dados. Escolhendo o melhor tipo de dados de cada coluna na sua tabela reduzirá o tamanho da tabela e fará operações nos dados bem mais rapidamente.

**P:** E então? São estes todos os tipos de dados?

**R:** Não, mas estes são os mais importantes. O tipo de dados DATE varia de acordo com o RDBMS, então será necessário consultar a tabela particular de documentação para mais informações. Nós recomendamos *SQL in a Nutshell* (O'Reilly) como um bom livro de referência que explica as diferenças entre sistemas RDBMS.

## QUE TIPO DE DADOS?

Determine que tipo de dados faz mais sentido para cada coluna. Enquanto fizer isso, preencha as outras informações pendentes.

O CEP em alguns países pode muitas vezes não ser de 8 caracteres, então usamos VARCHAR para poupar espaço no banco de dados. Você também pode ter usado CHAR neste campo levando em conta que o CEP deve ter um tamanho exato.

Nome da Coluna	Descrição	Exemplo	Melhor opção de Tipos de Dados
preco	O custo de um item à venda	5678.39	DEC(5,2)
cep	De 5 a 10 caracteres	90210-0010	VARCHAR(10) ←
peso_atomico	Peso atômico de um objeto com seis casas decimais	4.002602	DEC (10,6)
comentarios	Bloco de textos grande, para mais de 255 caracteres	Joe, estou na reunião de acionistas. Eles acabaram de fazer uma demonstração e há elásticos voando em direção à tela. Esta foi sua idéia de piada? Você precisa passar um tempo em Monster.com	BLOB
quantidade	A quantidade deste item disponível no estoque	239	INT
percentual_de_imposto	Uma porcentagem	3.755	DEC (4,2)
título_do_livro	Uma linha de texto	Use a cabeça SQL	VARCHAR (50)
sexo	Um caractere, M ou F	M	CHAR(1) { Um número de telefone será sempre do mesmo tamanho. Tratamos este campo como linha de texto }
numero_do_telefone	Dez dígitos, sem pontuação	2105552367	CHAR(10) { sempre do mesmo tamanho. Tratamos este campo como linha de texto }
estado	Abreviação em dois caracteres para um estado	RJ, SP	CHAR(2) { porque não precisamos realizar nenhuma operação matemática com eles, ainda que sejam números. }
aniversario	Dia, mês e ano	11/22/2006	DATE
jogos_vencidos	Um número inteiro representando o número de jogos vencidos	15	INT
hora_do_encontro	Um dia e hora	10:30 a.m. 4/12/2020	DATETIME

TIMESTAMP geralmente é utilizado para capturar a hora atual, DATETIME é a melhor opção em se tratando de armazenar um evento futuro



## PONTOS DE BALA

- Divida seus dados em categorias antes de criar sua tabela. Presta atenção, em especial, no tipo de dados para cada coluna.
- Use o comando **CREATE DATABASE** para criar o banco de dados que conterá todas as suas tabelas.
- Use o comando **USE DATABASE** para entrar no banco de dados a fim de criar suas tabelas.
- Todas as tabelas são criadas com o comando **CREATE TABLE**, contendo os nomes da coluna e os tipos de dados correspondentes.
- Alguns dos tipos de dados mais comuns são **CHAR**, **VARCHAR**, **BLOB**, **INT**, **DEC**, **DATE** e **DATETIME**. Cada um tem regras diferentes quanto a conteúdo.



o o

Espere um segundo. Onde está a tabela que acabei de criar no banco de dados `gregs_list`? Eu quero verificar se fiz tudo corretamente.

**Bem pensado. Verificar seu trabalho é o mais importante.**

Para verificar com o que se parece a tabela `meus_contatos` que criou, você pode usar o comando DESC para vê-la:

→ **DESC meus\_contatos;**

*DESC é a abreviação para DESCRIBE (descrever)*

Tente você agora.

File Edit Window Help DescTidy

> **DESC meus\_contatos;**

*Não se preocupe em relação a isto por enquanto; falaremos sobre isso em breve.*

## Sua tabela DESCrita

Quando estiver digitando o comando DESC, você verá algo similar a isto:

Column	Type	Null	Key	Default	Extra
sobrenome	varchar(30)	YES		NULL	
primeiro_nome	varchar(20)	YES		NULL	
email	varchar(50)	YES		NULL	
aniversario	data	YES		NULL	
profissao	varchar(50)	YES		NULL	
local	varchar(50)	YES		NULL	
estado_civil	varchar(20)	YES		NULL	
interesses	varchar(100)	YES		NULL	
procura	varchar(100)	YES		NULL	

9 rows in set (0.07 sec)



Eu queria ter colocado uma coluna "SEXO". É muito tarde para fazer isso?



### PODER DO CÉREBRO

O que você acha? Que tipo de problemas poderia surgir ao adicionar uma coluna?



## Ímã de geladeira - SQL

O código para criar um banco de dados e a tabela com a nova coluna "sexo" estão todos misturados em cima da geladeira. Você consegue reconstruir a linha de código para que eles funcionem? Alguns dos parênteses e ponto-e-vírgula caíram no chão e eram muito pequenos para serem encontrados, então fique à vontade para adicionar quantas achar necessário!

email VARCHAR(50)

aniversario DATE

USE gregs\_list

sobrenome VARCHAR(30)

primeiro\_nome VARCHAR(20)

interesses VARCHAR(100)

procura VARCHAR(100)

estado\_civil VARCHAR(20)

CREATE DATABASE gregs\_list

profissao VARCHAR(50)

local VARCHAR(50)

CREATE TABLE meus\_contatos

sexo CHAR(1)

**Ao terminar, tente digitar o novo código CREATE TABLE no seu sistema SQL para adicionar a coluna sexo!**



## Solução de Ímãs de geladeira - SQL

Seu trabalho era reconstruir a linha de código para organizar o código que criaria o banco de dados e a tabela com a nova coluna sexo.

**Você não pode recriar  
uma tabela ou banco  
de dados já existente!**

Você tentou digitar o novo comando `CREATE TABLE`? Se tentou, já deve saber que a solução deste exercício não ajudará a adicionar uma nova coluna.

Se tentou digitar o código em seu sistema SQL, você provavelmente viu algo assim:

```

CREATE DATABASE gregs_list
USE gregs_list
CREATE TABLE meus_contatos
(
    sobrenome VARCHAR(30),
    primeiro_nome VARCHAR(20),
    email VARCHAR(50),
    aniversario DATE,
    sexo CHAR(1),
    profissao VARCHAR(50),
    local VARCHAR(50),
    estado_civil ARCHAR(20),
    interesses VARCHAR(100),
    procura VARCHAR(100)
);
  
```

A nova coluna →  
para sexo

Oh, oh. Esse código gera  
uma mensagem de erro.  
Parece que a tabela não  
foi criada.

```

File Edit Window Help OhCrap!
> CREATE TABLE meus_contatos
-> (
->     sobrenome VARCHAR(30),
->     primeiro_nome VARCHAR(20),
->     email VARCHAR(50),
->     sexo CHAR(1),
->     aniversario DATE,
->     profissao VARCHAR(50),
->     local VARCHAR(50),
->     estado_civil VARCHAR(20),
->     interesses VARCHAR(100),
->     procura VARCHAR(100)
-> );
ERROR 1050 (42S01): Table 'meus_contatos' already exists
  
```

## Perguntas Idiotas

**P:** Quanto ao exercício Ímã de geladeira – SQL, por que obtivemos uma mensagem de erro?

**R:** Você não pode criar uma tabela que já existe. E, uma vez criado um banco de dados, não precisa criá-lo novamente. Outros erros possíveis estão em esquecer o ponto-e-vírgula. Verifique também se digitou qualquer uma das palavras-chave SQL.

**P:** Por que não há uma vírgula após “procura VARCHAR(100)” como todas as outras colunas têm?

**R:** A coluna “procura” é a última coluna antes de chegarmos aos parênteses, que indica ao sistema RDBMS que o final do comando é ali, assim sendo, a vírgula é desnecessária.

**P:** Então há um jeito de adicionar uma coluna esquecida, ou terei que refazer toda a tabela?

**R:** Você terá que refazê-la, mas antes de criar a tabela com a coluna sexo adicionada, terá que eliminar a tabela antiga. Como ainda não há dados na tabela, podemos simplesmente eliminar a tabela antiga e recomeçar a tabela.

**P:** E se eu já tenho uma tabela com dados inseridos e preciso adicionar uma coluna? Há uma maneira de fazer isso sem apagar a tabela inteira e começar tudo de novo?

**R:** Ótima pergunta! Há uma maneira de alterar sua tabela sem afetar os dados inseridos. Nós faremos isso mais adiante, mas no momento, como sua tabela está vazia, eliminaremos a tabela antiga e criaremos uma nova.



Se iremos digitar nosso comando CREATE TABLE tudo de novo, aposto que pouparíamos bastante tempo se digitássemos todos os nossos comandos SQL em um editor de texto tipo Bloco de Notas.

**Esta é uma ótima idéia, e você vai querer usar um editor de texto no decorrer deste livro.**  
Desta forma, poderá copiar e colar os comandos no seu sistema SQL quando quer que precise dele. Isto evitará que tenha que digitar tudo de novo. Você também pode copiar e editar comandos SQL antigos para criar comandos novos.

## Fora com a tabela antiga, pra dentro com a nova

- 1 Eliminar uma tabela é muito mais fácil que criar uma nova, use o simples comando:

O comando para apagar a tabela...

...e o nome da tabela a ser deletada.

**DROP TABLE meus\_contatos;**

→ Não se esqueça do ponto-e-vírgula.

File Edit Window Help ByeByeTable

```
> DROP TABLE meus_contatos;
Query OK, 0 rows affected (0.12 sec)
```

**DROP TABLE** funcionará independentemente da existência de dados na tabela, então use o comando com atenção extrema. Uma vez apagada, a tabela já era, assim como os dados inseridos nela.

**DROP TABLE**  
Apaga sua tabela e os dados inseridos nela!



Agora você pode digitar o novo comando **CREATE TABLE**:

```

File Edit Window Help Success
> CREATE TABLE meus_contatos
-> (
->     sobrenome VARCHAR(30) ,
->     primeiro_nome VARCHAR(20) ,
->     email VARCHAR(50) ,
->     sexo CHAR(1) ,
->     aniversario DATE ,
->     profissao VARCHAR(50) ,
->     local VARCHAR(50) ,
->     estado_civil VARCHAR(20) ,
->     interesses VARCHAR(100) ,
->     procura VARCHAR(100)
-> );
Query OK, 0 rows affected (0.05 sec)
  
```

*Desta vez → funcionou.*

Uma porção de palavras-chave e tipos de dados, com todos os disfarces, estão brincando o jogo "Quem sou eu?" Eles te dão uma pista, e você tenta adivinhar quem são eles, baseado no que dizem. Suponha que eles sempre dizem a verdade a respeito deles mesmos. Caso eles digam algo que possa ser verdade para mais de um comando, então escreva para todos aqueles para qual a frase se aplica. Preencha os espaços em branco próximos a frases com os nomes de um ou mais comandos.

Comandos de hoje::

**CREATE DATABASE, USE DATABASE, CREATE TABLE, DESC, DROP TABLE,  
CHAR, VARCHAR, BLOB, DATE, DATETIME, DEC, INT**

## Quem sou eu?



Nome

Eu tenho o seu número.

.....

Eu posso eliminar as tabelas indesejadas.

.....

Questões de V ou F são as minhas favoritas.

.....

Eu armazeno o aniversário de sua mãe.

.....

Eu tenho a tabela inteira em minhas mãos.

.....

Números são legais, mas eu odeio frações.

.....

Eu gosto de explicações longas e com muitas palavras.

.....

Esse é o lugar para armazenar tudo.

.....

A tabela não existiria sem mim.

.....

Eu sei exatamente quando será sua consulta no dentista semana que vem.

.....

Contadores gostam de mim.

.....

Eu posso melhorar o formato de sua tabela.

.....

Sem nós, você nem poderia criar a tabela.

.....

A  
pal  
INT

IN

Out  
Esta  
para

→ Respostas na página 45 .

## Anatomia de um Comando



Ok, eu já tenho minha nova tabela pronta. Agora, como colocar os dados dos meus papéis de anotações adesivas dentro da tabela?



## Para inserir dados na tabela, você usará o comando `INSERT`

Este comando faz essencialmente o que diz seu nome. Dê uma olhada no comando abaixo para ver como cada parte funciona. Os valores no segundo conjunto de parênteses **devem estar na mesma ordem que os nomes das colunas**.

O comando abaixo não é um comando real, é um exemplo para mostrar o formato do comando `INSERT`.

```
?
? A palavra-chave INSERT INTO inicia a declaração
? O nome da sua tabela, no caso do Greg, será meus_contatos.
? Esta próxima parte de nomes das suas colunas, separadas por vírgulas. Você já sabe que a lista de Greg terá colunas como nome, sobrenome, primeiro_nome e email.
? Mais nomes de colunas após a última.
? Mais valores seguem, não há vírgula no último valor.
? O ponto-e-vírgula de costume encerrando o código.
? IMPORTANTE: Os valores precisam estar na mesma ordem que os nomes das colunas.
```

`INSERT INTO your_table (column_name1, column_name2, ...)`

`VALUES ('value1', 'value2', ...);`

Outra palavra-chave. Esta indica que os valores para as colunas a seguem.

Essa próxima parte é uma lista dos valores, separadas por vírgulas. No caso do Greg, a lista conterá as informações das anotações adesivas.

## QUE TIPO DE DADOS?

Antes de escrever o comando INSERT, você precisa combinar os nomes das colunas com seus respectivos valores.

Colunas	Valores
<code>primeiro_nome</code>	'Relacionamento, Amigos'
<code>estado_civil</code>	'Anderson'
<code>procura</code>	'05/09/1980'
<code>sexo</code>	'Escritora Técnica'
<code>aniversario</code>	'Jillian'
<code>sobrenome</code>	'Solteiro'
<code>local</code>	'F'
<code>interesses</code>	'Palo Alto, CA'
<code>profissao</code>	'jill_anderson@breakneckpizza.com'
<code>email</code>	'Caiaque, Répteis'

## QUE TIPO DE DADOS?

Antes de escrever o comando INSERT, você precisa combinar os nomes das colunas com seus respectivos valores.

Colunas	Valores
<code>primeiro_nome</code>	'Relacionamento, Amigos'
<code>estado_civil</code>	'Anderson'
<code>procura</code>	'05/09/1980'
<code>sexo</code>	'Escritora Técnica'
<code>aniversario</code>	'Jillian'
<code>sobrenome</code>	'Solteiro'
<code>local</code>	'F'
<code>interesses</code>	'Palo Alto, CA'
<code>profissao</code>	'jill_anderson@breakneckpizza.com'
<code>email</code>	'Caiaque, Répteis'

*O tipo DATE requer um formato específico. Verifique sua documentação SQL por especificações.*

*Não esqueça! Você precisa de aspas simples em valores de caractere único*

## Criando uma declaração INSERT

Os nomes de suas colunas estão no primeiro conjunto de parênteses e divididos por vírgulas.

Você pode apertar ENTER antes de abrir um novo parêntese para tornar o código mais fácil para ler no seu sistema SQL.

**INSERTO INTO meus\_contatos**

(sobrenome, primeiro\_nome, email, sexo, aniversario, profissao, local, estado\_civil, interesses e procura)

**VALUES**

Aperte ENTER após fechar os parentes da coluna e outros VALUES para fazer o código de fácil leitura.

('Anderson', 'Jillian', 'jill\_anderson@breakneckpizza.com', 'F', '05/09/1980', 'Escritora Técnica', 'Palo Alto, CA', 'Solteiro', 'Caiaque, Répteis, Relacionamentos, Amigos');

Os valores para cada coluna estão no segundo conjunto de parênteses e, desta vez, separados por vírgulas.

Qualquer valor que possa ir em uma coluna VARCHAR, CHAR, DATE ou BLOB deve ficar entre aspas.



**Veja Isto!**

A ordem importa!

Os valores deveriam ser listados exatamente na mesma ordem que os nomes das colunas.



### Tente em casa

Esta é uma das formas de adicionar linhas à sua tabela. Tente digitar para você mesmo. Primeiramente, digite em um editor de texto para que no caso de enganos não tenha que redigitar sua tabela. Presta bastante atenção com as aspas simples e vírgulas. Escreva as respostas aqui:

Exercícios



Você acabou de me dizer que valores em CHAR, VARCHAR, DATE e BLOB usam aspas simples em volta delas no comando INSERT. Isto quer dizer que valores numéricos como DEC e INT não usam aspas?

### Exatamente certo.

Aqui está um comando INSERT que você poderá usar se você tivesse uma tabela de compras de donuts. Perceba como, em valores, os números que acompanham as dúzias de donuts comprados e as columnas de preço não possuem aspas.

A coluna de dúzias é um INT, uma vez que as vendas nunca são feitas em frações de dúzias e por não se utilizar números decimais.

A coluna de preço é DEC(4,2), o que significa ter o tamanho de 4 caracteres, com duas casas decimais.

```
INSERT INTO compra_de_donuts  
(tipo_donut, duzias, cobertura, preco, valor)  
VALUES
```

```
('jelly', 3, 'confeitos', 10.50);
```

Os valores inseridos na coluna de dúzias e na coluna de preço não precisam de aspas.



Aponte seu lápis

Seu sistema SQL RDBMS dirá a você quando algo der errado no seu código, mas às vezes será meio vago. Dê uma olhada em cada comando INSERT abaixo. Primeiro tente adivinhar o que há de errado com o comando, daí então tente digitá-lo no sistema e observar como o sistema responderá.

`INSERT INTO meus_contatos`

`(sobrenome, primeiro_nome, email, sexo, aniversario, profissao, local, estado_civil, interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com', 'F', '05-09-1980', 'Escritora Técnica', 'Solteira', 'Caiaque, Répteis', 'Relacionamento, Amigos');`

O que está errado?

Seu sistema SQL diz:

`INSERT INTO meus_contatos`

`(sobrenome, primeiro_nome, sexo, aniversario, profissao, local, estado_civil, interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com', 'F', '05-09-1980', 'Escritora Técnica' 'Palo Alto, CA', 'Solteira', 'Caiaque, Répteis', 'Relacionamento, Amigos');`

O que está errado?

Seu sistema SQL diz:

`INSERT INTO meus_contatos`

`(sobrenome, primeiro_nome, email, sexo, aniversario, profissao, local, estado_civil, interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com', 'F', '05-09-1980', 'Escritora Técnica' 'Palo Alto, CA', 'Solteira', 'Caiaque, Répteis', 'Relacionamento, Amigos');`

O que está errado?

Seu sistema SQL diz:

`INSERT INTO meus_contatos`

`(sobrenome, primeiro_nome, email, sexo, aniversario, profissao, local, estado_civil, interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com', 'F', '05-09-1980', 'Escritora Técnica', 'Palo Alto, CA', 'Solteira', 'Caiaque, Répteis', 'Relacionamento, Amigos');`

O que está errado?

Seu sistema SQL diz:

*Se neste comando seu sistema demorar, tente digitar uma aspa simples seguida de um ponto-e-vírgula após ter digitado o restante do comando.*



## Aponte seu lápis

### Solução

Var

Há trê

Seu sistema SQL RDBMS dirá a você quando algo der errado no seu código, mas às vezes será meio vago. Dê uma olhada em cada comando INSERT abaixo. Primeiro tente adivinhar o que há de errado com o comando, daí então tente digitá-lo no sistema e observar como o sistema responderá.

```
INSERT INTO meus_contatos
```

```
(sobrenome, primeiro_nome, email, sexo, aniversario, profissao, local, estado_civil,
interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.
com', 'F', '05-09-1980', 'Escritora Técnica', 'Solteira', 'caiaque, Répteis',
'Relacionamento, Amigos');
```

O que está errado? *Está faltando um valor para local*

*Temos uma coluna local na lista de colunas, mas não temos  
nenhum valor inserido para local na lista de valores, nos  
não tínhamos um dos valores*

Seu sistema SQL diz: *ERROR 1136 (21501): Column count doesn't match value count at row!*

```
INSERT INTO meus_contatos
```

```
(sobrenome, primeiro nome, sexo, aniversario, profissao, local, estado_civil,
interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com',
'F', '05-09-1980', 'Escritora Técnica', 'Palo Alto, CA', 'Solteira', 'Caiaque, Répteis',
'Relacionamento, Amigos');
```

O que está errado? *Está faltando a coluna email, na lista de colunas.*

*Perceba que diferentes problemas resultam em um mesmo  
aviso de erro. Fique atento com os tipos; eles podem ser  
meio difíceis de serem rastreados.*

*Desta vez, nós temos um valor para cada coluna, mas  
não temos a coluna email na lista de colunas*

Seu sistema SQL diz: *ERROR 1136 (21501): Column count doesn't match value count at row!*

```
INSERT INTO meus_contatos
```

```
(sobrenome, primeiro nome, email, sexo, aniversario, profissao, local, estado_civil,
interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com',
'F', '05-09-1980', 'Escritora Técnica', 'Palo Alto, CA', 'Solteira', 'Caiaque, Répteis',
'Relacionamento, Amigos');
```

O que está errado? *Está faltando uma vírgula entre dois valores.*

*Não há vírgula na lista de valores entre  
Escritora Técnica e Palo Alto, CA*

Seu sistema SQL diz: *ERROR 1136 (21501): Column count doesn't match value count at row!*

```
INSERT INTO meus_contatos
```

```
(sobrenome, primeiro nome, email, sexo, aniversario, profissao, local, estado_civil,
interesses, procura) VALUES ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com',
'F', '05-09-1980', 'Escritora Técnica', 'Palo Alto, CA', 'Solteira', 'Caiaque, Répteis',
'Relacionamento, Amigos');
```

O que está errado? *Está faltando uma aspa simples no final do último valor inserido.*

Seu sistema SQL diz: *ERROR 1064 (42000): You have an error in your SQL syntax; check*

*the manual that corresponds to your MySQL server version for the  
right syntax to use near at line 4*

# Variações para o comando INSERT

Há três variações para o comando INSERT que você deve saber:

## 1 Mudando a ordem das colunas

Você pode mudar a ordem das colunas, desde que os respectivos valores para cada coluna venham na mesma ordem!

```
INSERT INTO meus_contatos
  (interesses, primeiro_nome, sobrenome, sexo, email, aniversario,
  profissao, local, estado_civil, procura)
VALUES
  ('Caiaque, Répteis', 'Jillian', 'Anderson', 'F', 'jill_anderson@breakneckpizza.com', '05-09-1980', 'Escritora Técnica', 'Palo Alto, CA', 'Solteira', 'Relacionamento, Amigos');
```

Percebeu a ordem do nome das colunas? Agora olhe para os valores; eles estão naquela mesma ordem. Contanto que os valores sejam referentes aos respectivos nomes de colunas, a ordem que usar o INSERT não importará para você, no seu sistema SQL!

## 2 Omitindo nomes de colunas

Você pode omitir a lista de nomes de colunas, mas os valores devem estar **todos ali**, e na **mesma ordem** que **adicionou as colunas**  
(Cheque duas vezes a ordem na página 33, caso não tenha certeza.)

```
INSERT INTO meus_contatos
  VALUES
    ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com',
    'F', '05-09-1980', 'Escritora Técnica', 'Palo Alto, CA',
    'Solteira', 'Caiaque, Répteis', 'Relacionamento, Amigos');
```

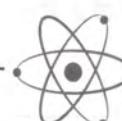
Nós deixamos de colocar os nomes das colunas, mas se você fizer isso, deverá incluir TODOS os valores, e na ORDENA EXATA em que elas estejam na tabela!

## 3 Deixando algumas colunas de fora

Você pode inserir algumas colunas e deixar outras de fora.

```
INSERT INTO meus_contatos
  (sobrenome, primeiro_nome, email)
VALUES
  ('Anderson', 'Jillian', 'jill_anderson@breakneckpizza.com');
```

Desta vez, estamos inserindo somente parte dos nossos dados. Como o seu sistema SQL não saberá que partes são estas, você deve especificá-las na lista de colunas e os valores que você está inserindo.

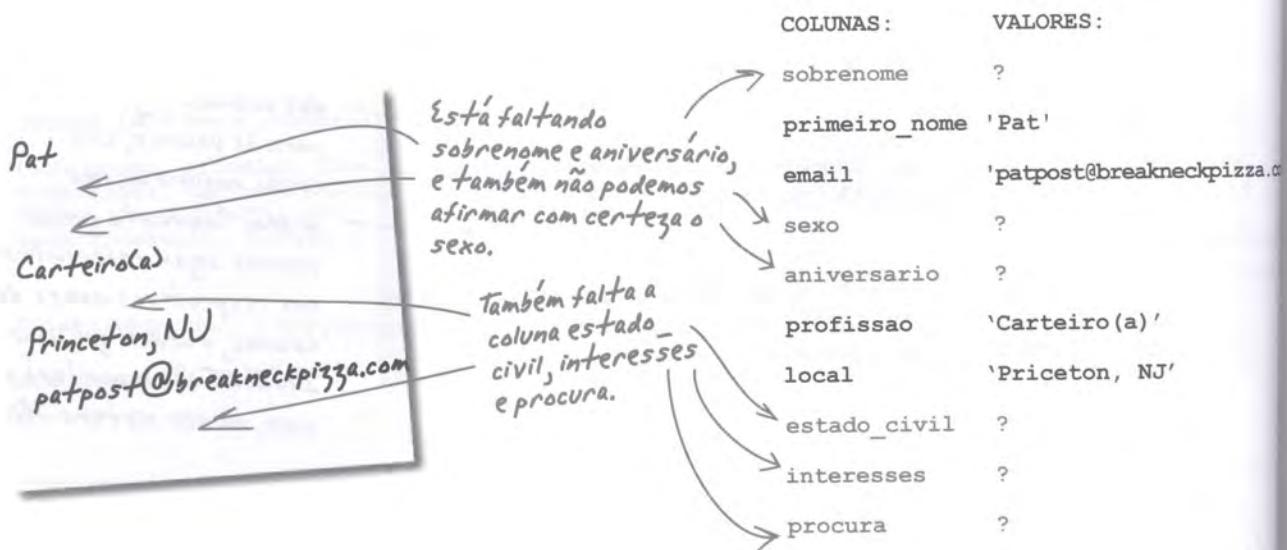


### PODER DO CÉREBRO

O que você acha que aparece naquelas colunas em que não inseriu valor algum?

## Colunas sem valores

Vamos inserir uma informação no banco de dados `meus_contatos` a partir desta anotação adesiva incompleta:



Pelo fato de a anotação adesiva estar incompleta, Greg terá que inserir um registro incompleto. Mas tudo bem, ele estará apto a inserir as informações pendentes em momento posterior.

Estamos usando uma versão de `INSERT` onde não temos que fornecer dados para todas as colunas porque nos permite incluir apenas as colunas onde sabemos os valores.

```

    INSERT INTO meus_contatos
    (primeiro_nome, email, profissao, local)
    VALUES
    ('Pat', 'patpost@breakneckpizza.com',
    'Carteiro(a)', 'Princeton, NJ');
  
```

```

File Edit Window Help MoreDataPlease
> INSERT INTO meus_contatos (primeiro_nome, email, profissao,
local) VALUES ('Pat', 'patpost@breakneckpizza.com',
'Carteiro(a)', 'Princeton, NJ');

Query OK, 1 row affected (0.02 sec)
  
```

## Espie sua tabela com o comando SELECT

Então você quer ver com o que sua tabela irá se parecer? Bem, DESC não vai funcionar mais, porque ela só mostra a estrutura da tabela e não a informação dentro dela. Sendo assim, você deveria usar um simples comando SELECT para que possa ver os dados dentro dela.

	sobrenome	primeiro_nome	email	aniversario	sexo	local	interesses	proissao	estado_civil	procura
1	NULL	Pat	patpost@breakneckpizza.com	NULL	NULL	Princeton, NJ	NULL	NULL	NULL	Há um valor NULL de cada coluna.
2	Anderson	Jillian	jill_anderson@breakneckpizza.com	1980-09-05	F	Escritora Técnica	Palo Alto, CA	Solteira	Caiaque, Réptilos	Amigos, Relacionamento, amigos

2 rows in set (0.00 sec)

Nós queremos ver todos os dados de nossa tabela...

...e o asterisco diz para selecionar tudo.

O nome da nossa tabela.

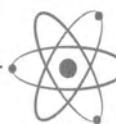
**SELECT \* FROM meus\_contatos;**



No momento, não se preocupe como que o comando SELECT faz,

Veremos com muito mais detalhe este comando no capítulo 2. Por ora, sente-se e maravilhe-se com a beleza de sua tabela ao usar este comando.

Agora tente você mesmo. Maximize sua janela para ver o resultado com qualidade.



**PODER DO CÉREBRO**

Agora você sabe que NULL aparece em toda coluna sem valor inserido. O que você acha que NULL realmente significa?



## SQL exposto

Entrevista da Semana:  
Confissões de um NULL

**Use a cabeça:** Bem-vindo, NULL. Tenho que admitir que não esperava vê-lo. Nunca pensei que você de fato existisse. O boato é que você não é nada mais que um zero à esquerda.

**NULL:** Não acredito que tenha dado ouvidos a tais mentiras. Sim, estou aqui, e sou bem real! Então você pensa que eu não sou nada, né? Só poeira embaixo de seus sapatos?

**Use a cabeça:** Opa, fique calmo. Isso é porque você só aparece quando algo não tem valor...

**NULL:** Claro, melhor do que um zero ou uma linha vazia.

**Use a cabeça:** O que é uma linha vazia?

**NULL:** Seria o uso de duas aspas simples com nada dentro. Ainda é um campo de texto, mas de tamanho zero. É como definir para `primeiro_nome` na tabela `meus_contatos` o valor "".

**Use a cabeça:** Então você não é apenas um jeito chique pra dizer nada?

**NULL:** Eu já te disse! Eu não sou nada! Eu sou algo, só que... um pouco indefinido, é tudo.

**Use a cabeça:** Quer dizer que seu eu te comparar a um zero ou uma linha vazia você não se igualaria?

**NULL:** Não! Eu nunca me igualaria a um zero. E, na verdade, não me igualaria nem a outro NULL. Você não pode comparar um NULL com outro. Um valor pode ser NULL, mas nunca igual a NULL porque NULL é um valor indefinido! Entende?

**Use a cabeça:** Acalme-se e me esclareça. Você não é igual a zero, não é uma linha vazia variável, e não é igual nem a você mesmo? Isto não faz sentido!

**NULL:** Sei que isso é confuso. Apenas pense em mim da seguinte forma: Eu sou indefinido. Sou como o interior de uma caixa nunca aberta.. Qualquer coisa poderia estar ali dentro, então você não pode comparar uma caixa não aberta com outra porque você não sabe o que terá dentro de cada uma. Poderei até estar vazio, mas não saberá.

**Use a cabeça:** Eu tenho ouvido rumores de que você não é estimado. Que há momentos que talvez vocês NULLs causem problemas.

**NULL:** Eu tenho que admitir que já apareci em lugares onde era indesejado. Algumas colunas deveriam estar sempre com valores. Como sobrenomes, por exemplo. Não há razão em ter um NULL como sobrenome em uma tabela.

**Use a cabeça:** Então você não iria onde não fosse chamado?

**NULL:** Correto! Agora me diga, quando estiver construindo sua tabela e criando suas colunas, me avise.

**Use a cabeça:** Você não parece muito com uma caixa não aberta.

**NULL:** Já basta. Eu tenho lugares para ir, valores para me tornar.

## Controlando seu NULL interior

Há algumas colunas na sua tabela que deveriam sempre estar com valores. Ainda lembra da anotação adesiva incompleta para Pat, com nenhum sobrenome? Ela (ou ele) não será muito fácil de ser encontrada quando tiver mais vinte sobrenomes NULL inseridos em sua tabela. Poderá facilmente ajustar sua tabela para não aceitar valores NULL para colunas.

```
CREATE TABLE meus_contatos
```

```
(
```

```
    sobrenome VARCHAR (30) NOT NULL,
```

```
    primeiro_nome VARCHAR (20) NOT NULL
```

```
);
```

Apenas adiciona as palavras `NOT NULL` após o tipo de dados.

Se usá-las, forneça um valor para a coluna, no comando `INSERT`. Se não o fizer, vai obter uma mensagem de erro.



Aponte seu lápis

```
CREATE TABLE meus_contatos
(
    sobrenome VARCHAR (30) NOT NULL,
    primeiro_nome VARCHAR (20) NOT NULL,
    email VARCHAR(50),
    sexo CHAR(1),
    aniversario DATE,
    profissao VARCHAR(50),
    local VARCHAR(50),
    estado_civil VARCHAR(20),
    interesses VARCHAR(100),
    procura VARCHAR(100)
);
```

Olhe para cada coluna no seu comando CREATE TABLE para meus\_contatos. Quais deveriam ser ajustadas para opção NOT NULL? Pense naquelas colunas que nunca deveriam ser NULL e as circule.

Nós demos duas para começar, agora termine o resto. Primeiramente, considere colunas que usará posteriormente para fazer pesquisa ou colunas que são únicas.



Aponte seu lápis

Solução

```
CREATE TABLE meus_contatos
(
    sobrenome VARCHAR (30) NOT NULL,
    primeiro_nome VARCHAR (20) NOT NULL,
    email VARCHAR(50),
    sexo CHAR(1),
    aniversario DATE,
    profissao VARCHAR(50),
    local VARCHAR(50),
    estado_civil VARCHAR(20),
    interesses VARCHAR(100),
    procura VARCHAR(100)
);
```

Olhe para cada coluna no seu comando CREATE TABLE para meus\_contatos. Quais deveriam ser ajustadas para opção NOT NULL? Pense naquelas colunas que nunca deveriam ser NULL e as circule.

Nós demos duas para começar, agora termine o resto. Primeiramente, considere colunas que usará posteriormente para fazer pesquisa ou colunas que são únicas.

*Todas as colunas deveriam ser NOTNULL*

*Você utilizará TODAS as suas colunas para fazer pesquisa. É importante saber que seus registros estão completos e sua tabela tem dados completos inseridos...*

*...mas, se você tem uma coluna que saberá que tem que ser preenchida posteriormente, considere permitir o uso de valores NULL nela.*

# NOT NULL aparece em DESC

Aqui está como a tabela meus\_contatos vai parecer se você ajustar todas as colunas para terem valores NOT NULL.

Pree

Se você  
(DEFA  
é inseri

```

File Edit Window Help NoMoreNULLs
CREATE TABLE meus_contatos
(
    sobrenome VARCHAR (30) NOT NULL,
    primeiro_nome VARCHAR (20) NOT NULL,
    email VARCHAR(50) NOT NULL,
    aniversario DATE NOT NULL,
    profissao VARCHAR(50) NOT NULL,
    local VARCHAR(50) NOT NULL,
    estado_civil VARCHAR(20) NOT NULL,
    interesses VARCHAR(100) NOT NULL,
    procura VARCHAR(100) NOT NULL
);
Query Ok, 0 rows affected (0.01 sec)

> DESC meus_contatos;
+-----+-----+-----+-----+-----+-----+
| Column      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| sobrenome   | varchar(30) | NO  |   |   |   |
| primeiro_nome | varchar(20) | NO  |   |   |   |
| email       | varchar(50) | NO  |   |   |   |
| sexo         | char(1)    | NO  |   |   |   |
| aniversario | date       | NO  |   |   |   |
| profissao   | varchar(50) | NO  |   |   |   |
| local        | varchar(50) | NO  |   |   |   |
| estado_civil | varchar(20) | NO  |   |   |   |
| interesses  | varchar(100) | NO  |   |   |   |
| procura     | varchar(100) | NO  |   |   |   |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.02 sec)

```

*Aqui está onde criamos nossa tabela com NOTNULL em cada coluna.*

*Aqui está a tabela descrita. Observe a palavra NO abaixo da coluna NULL.*

*Aqui e  
da col  
os do*

## Preencha os espaços em branco com DEFAULT

Se você tivesse uma coluna que sabe que comumente tem um valor específico, nós podemos lhe atribuir um valor padrão (**DEFAULT**). O valor que segue a palavra-chave **DEFAULT** é automaticamente inserido na tabela cada vez que uma linha é inserida *caso nenhum outro valor seja atribuído*. O valor padrão deve ser do mesmo tipo de valor que a coluna.

```
CREATE TABLE lista_donut
(
    nome_donut VARCHAR(10) NOT NULL,
    tipo_donut VARCHAR(6) NOT NULL,
    custo_donut DEC(3,2) NOT NULL DEFAULT 1.00
);
```

Queremos ter certeza que sempre haverá um valor nesta coluna.  
Podemos não só fazer a NOT NULL como também lhe atribuir um valor DEFAULT de \$1.

nome_donut	tipo_donut	custo_donut
Blooberry	recheado	2.00
Cinnamondo	anel	1.00
Rockstar	enrolado	1.00
Carameller	enrolado	1.00
Appleblush	recheado	1.40

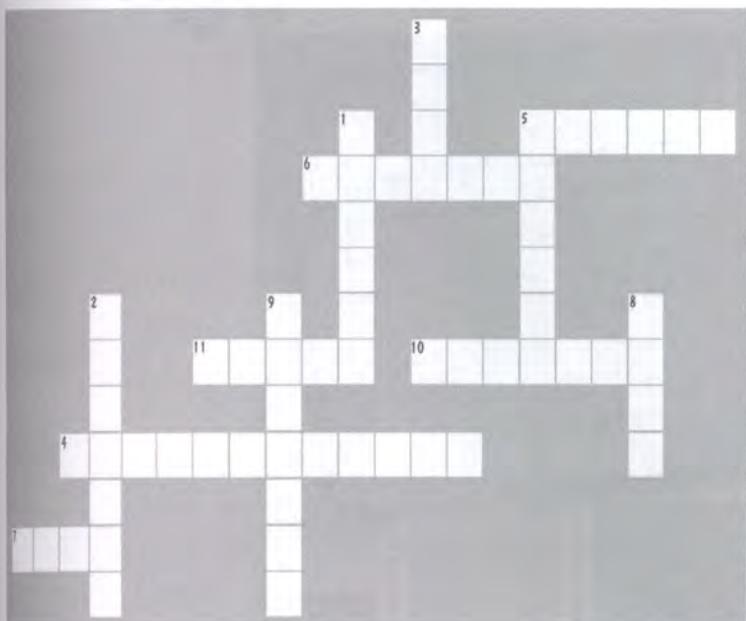
Este será o valor inserido na tabela para a coluna **custo\_donut** quando nenhum outro valor for designado.

Aqui está como sua tabela seria se você deixasse os valores da coluna **custo\_donut** em branco ao inserir os registros para os donuts Cinnamondo, Rockstar e Carameller.



## Palavras cruzadas

Reserve um tempinho para relaxar e dê ao lado esquerdo do seu cérebro algo para fazer. São palavras cruzadas comuns; todas as respostas estão neste capítulo.



### Horizontais

- Um \_\_\_\_\_ é um container que guarda todas as tabelas e outras estruturas SQL relacionadas a estas tabelas.
- Uma \_\_\_\_\_ é um tipo ou porção de dados armazenado pela sua tabela.
- Este armazena dados em texto de até 255 caracteres de tamanho.
- Você não pode comparar um \_\_\_\_\_ com outro.
- Este é um conjunto único de colunas para descrever atribuições de um objeto.
- Para adicionar dados na sua tabela, você usará o comando \_\_\_\_\_.

### Verticais

- É uma estrutura interna de seu banco de dados que contém dados, organizados em colunas e linhas.
- Use-o em seu comando CREATE TABLE para especificar um valor para uma coluna se nenhum outro valor for inserido.
- Use esta palavra para ver a tabela que acabou de criar.
- Esta palavra pode ser usada na frente de ambos TABLE ou DATABASE.
- Para eliminar sua tabela use \_\_\_\_\_ TABLE.
- Este campo datatype (tipo de dados) pensa que deveria ser inteiro, mas não tem medo de números negativos.



## Sua caixa de ferramenta SQL

Você já tem o capítulo 1 na palma da mão, e já sabe como criar banco de dados e tabelas, como também inserir alguns dos mais comuns tipos de dados e tabelas, bem como inserir alguns de seus dados mais comuns enquanto se certifica quais colunas precisam de um valor.

**CREATE DATABASE**  
Use este comando para ajustar o banco de dados que conterá todas as suas tabelas.

**USE DATABASE**  
Leva você ao interior do banco de dados para ajustar todas as suas tabelas.

**CREATE TABLE**  
Inicia ajustando sua tabela, mas você precisará também saber o nome de suas colunas e tipos de dados. Você deve ter trabalhado nisto ao analisar o tipo de dados que estará colocando em sua tabela.

**NULL and NOT NULL**  
Você precisará também ter uma ideia de quais tabelas não utilizarão os valores NULL para ajudá-lo a pesquisar posteriormente seus dados. Precisará ajustar as colunas para NOT NULL quando criar sua tabela.

**DEFAULT**  
Permite que você especifique valor padrão para uma coluna, usado se você não fornecer um valor para a coluna ao inserir um registro.

**DROP TABLE**  
Permite que você apague toda a tabela caso tenha cometido algum equívoco, mas só poderá fazer isso antes de usar o comando INSERT, que lhe permite adicionar valores para cada coluna.



### PONTOS DE BALA

- Se quer ver a estrutura de sua tabela, use o comando **DESC**.
- O comando **DROP TABLE** pode ser usado para jogar sua tabela fora. Use com cuidado!
- Para inserir seus dados dentro da tabela, use uma das variações do comando **INSERT**.
- Um valor **NULL** é um valor indefinido. Ele não se iguala a zero ou a valor vazio. Uma coluna com um valor **NULL** é **NULL**, mas nunca igual a outro **NULL**.
- Colunas que não tiveram valores atribuídos a ela através do comando **INSERT** são ajustadas para **NULL** por padrão.
- Você pode alterar uma tabela, fazendo-a não aceitar um valor **NULL** ao usar as palavras-chave **NOT NULL** quando criar a tabela.
- Usar um valor **DEFAULT** quando você **CREATE** (criar) sua tabela, a coluna é preenchida com aquele valor, caso tenha inserido um registro sem valor algum para aquela coluna.





## 2 O comando SELECT

### Selecionando Dados

SELECT\*FROM presentes  
WHERE conteúdo = \*caro\*;



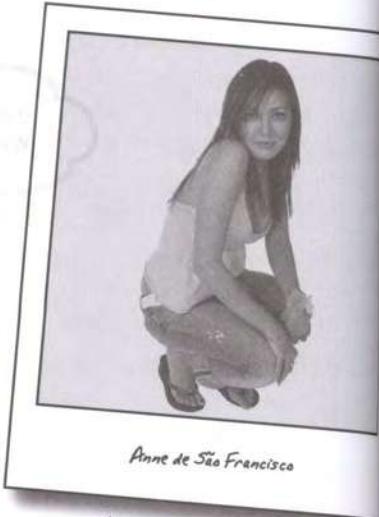
**Fala sério, é melhor acessar do que restaurar?** Quando o assunto é bancos de dados, as chances são de que você terá que acessar os seus dados na mesma freqüência que as insere. É aí que este capítulo entra: você conhecerá o poderoso comando SELECT e aprender como ganhar acesso àquela importante informação que tem inserido nas tabelas. E ainda vai aprender como usar WHERE, AND e OR para acessar suas tabelas e ainda evitar a exibição de dados de que não precisa.

## Marcar ou não marcar um encontro?

Greg já terminou de inserir todas as suas anotações contidas nos adesivos na tabela `meus_contatos`. Agora ele está pronto para relaxar. Ele tem dois ingressos para um show e ele quer convidar um de seus contatos, uma garota de São Francisco para um encontro.

Ele precisa encontrar o email dela, então ele usa o comando SELECT do capítulo 1 para visualizar sua tabela.

```
SELECT * from meus_contatos;
```



Os detalhes dela estão  
na tabela do Greg...em  
algum lugar.

### Seja o Greg

Seu trabalho é interpretar Greg. Procure ao longo da primeira parte da tabela `meus_contatos` na próxima página por Anne de São Francisco.



A tabela meus\_contatos tem uma  
porção de colunas. Estas são só as  
primeiras.

File Edit Window Help AnneWho

sobrenome	primeiro_nome	email	sexo
Anderson	Jillian	jill_anderson@breakneckpizza.com	F
Joffe	Kevin	joffe@simuduck.com	M
Newsome	Amanda	aman2luv@breakneckpizza.com	F
Garcia	Ed	ed99@b0tt0msup.com	M
Roundtree	Jo-Ann	jojoround@breakneckpizza.com	F
Briggs	Chris	cbriggs@boards-r-us.com	M
Harte	Lloyd	hovercraft@breakneckpizza.com	M
Toth	Anne	Anne_Toth@leapinlimos.com	F
Wiley	Andrew	andrewwiley@objectville.net	M
Palumbo	Tom	palofmine@mightygumball.net	M
Ryan	Alanna	angrypirate@breakneckpizza.com	F
McKinney	Clay	clay@starbuzzcoffee.com	M
Meeker	Ann	ann_meeker@chocoholic-inc.com	F
Powers	Brian	bp@honey-doit.com	M
Manson	Anne	am86@objectville.net	M
Mandel	Debra	debmونster@breakneckpizza.com	F
Tedesco	Janis	janistedesco@starbuzzcoffee.com	F
Talwar	Vikram	vikt@starbuzzcoffee.com	M
Szwed	Joe	szwed_joe@objectville.net	M
Sheridan	Diana	sheridi@mightygumball.net	F
Snow	Edward	snowman@tikibeanlounge.com	M
Otto	Glenn	glenn0098@objectville.net	M
Hardy	Anne	anneh@b0tt0msup.com	F
Deal	Mary	nobigdeal@starbuzzcoffee.com	F
Jagel	Ann	dreamgirl@breakneckpizza.com	F
Melfi	James	dirmelfi@b0tt0msup.com	M
Oliver	Lee	lee_oliver@weatherorama.com	M
Parker	Anne	annep@starbuzzcoffee.com	F
Ricci	Peter	ricciman@tikibeanlounge.com	M
Reno	Grace	grace23@objectville.net	F
Moss	Zelda	zelda@weatherorama.com	F
Day	Clifford	cliffnight@breakneckpizza.com	M
Bolger	Joyce	joyce@chocoholic-inc.com	F
Blunt	Anne	anneblunt@breakneckpizza.com	F
Bolling	Lindy	lindy@tikibeanlounge.com	F
Gares	Fred	fgares@objectville.net	M
Jacobs	Anne	anne99@objectville.net	F
Ingram	Dan	danielingram@breakneckpizza.com	M

local
Palo Alto, CA
San Jose, CA
San Fran, CA
San Mateo, CA
San Fran, CA
Austin, TX
San Jose, CA
San Fran, CA
NYC, NY
Princeton, NJ
San Fran, CA
NYC, NY
San Fran, CA
Napa, CA
Seattle, WA
Natchez, MS
Las Vegas, NV
Palo Alto, CA
NYC, NY
Phoenix, AZ
Fargo, ND
Boulder, CO
San Fran, CA
Boston, MA
San Fran, CA
Dallas, TX
St. Louis, MO
San Fran, CA
Reno, NV
Palo Alto, CA
Sunnyvale, CA
Chester, NJ
Austin, TX
San Fran, CA
San Diego, CA
San Jose, CA
San Jose, CA
Miami, FL

este não é o fim da tabela, Greg  
tinha um MONTÉ de anotações  
adesivas.



## Solução do "Seja o Greg"

Seu trabalho era interpretar Greg procurando na primeira parte da tabela `meus_contatos` por Anne de São Francisco.

Você tinha que encontrar todas as Anne de São Francisco e anotar o nome e o sobrenome delas, bem como o endereço de email.

Toth, Anne: [Anne\\_Toth@leapinlimos.com](mailto:Anne_Toth@leapinlimos.com)

Hardy, Anne: [anneh@bottomsup.com](mailto:anneh@bottomsup.com)

Parker, Anne: [annep@starbuzzcoffee.com](mailto:annep@starbuzzcoffee.com)

Blunt, Anne: [anneblunt@breakneckpizza.com](mailto:anneblunt@breakneckpizza.com)

*Aqui estão todas as Anne e seus respectivos emails.*

*Greg está procurando por Anne, com e no final. Caso tenha encontrado algum registro de Ann, você pode ter ignorado*

## Fazendo contato

Aqui **tomou tempo demais e foi extremamente tedioso**. Há uma grande possibilidade de o Greg ter passado desapercebido por algumas das Annes, incluindo aquela que ele procurava.

Agora que ele tem o email de todas elas, é só mandar e descobrir...

Para: Blunt, Anne <[anneblunt@breakneckpizza.com](mailto:anneblunt@breakneckpizza.com)>  
De: Greg <[greg@gregslist.com](mailto:greg@gregslist.com)>  
Assunto: Nós nos encontramos no Starbuzz?

Eu tenho procurado por um vaqueiro como você!  
Encontre-me às cinco horas e iremos levantar po-

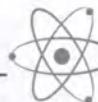
Para: Tosh, Anne <[Anne\\_toth@leapinlimos.com](mailto:Anne_toth@leapinlimos.com)>  
De: Greg <[greg@gregslist.com](mailto:greg@gregslist.com)>  
Assunto: Nós nos encontramos no Starbuzz?

mento estou com um cara maravilhoso  
que não conhecemos em  
Para: Hardy, Anne <[anne@b0t00msup.com](mailto:anne@b0t00msup.com)>  
De: Greg <[greg@gregslist.com](mailto:greg@gregslist.com)>  
Assunto: Nós nos encontramos no Starbuzz?

Eu não sou a Anne que você está procurando, mas  
tenho certeza que ela deve ser um doce de garota  
Se as coi-

Para: Parker, Anne <[annep@starbuzzcoffee.com](mailto:annep@starbuzzcoffee.com)>  
De: Greg <[greg@gregslist.com](mailto:greg@gregslist.com)>  
Assunto: Nós nos encontramos no Starbuzz?

Mas claro que lembro de você! Eu gostaria que você  
tivesse me ligado antes. Fiz planos com meu ex-  
namorado, que quer voltar a namorar comigo.



### PODER DO CÉREBRO

Você consegue pensar em um jeito de escrevermos uma consulta para exibir apenas os registros de `primeiro_nome` inseridos como "Anne"?

## Um SELECT melhor

Aqui está um comando SELECT que teria ajudado Greg a encontrar Anne muito mais rápido que ler detalhadamente toda aquele enorme tabela procurando por ela. No comando, usamos uma **cláusula WHERE que dá ao sistema SQL algo específico para procurar**. Ela limita os resultados para nós e **exibe somente as linhas que são compatíveis com a condição estabelecida**.

O sinal de igual na cláusula WHERE para testar se cada valor na coluna `primeiro_nome` é igual ou compatível ao texto 'Anne'. Se for compatível, todos os valores daquela linha são exibidos. Se não, a linha não é exibida.

```

File Edit Wi
> SELEC
+-----+
| sobre
+-----+
| Toth
| Manso
| Hardy
| Parke
| Blunt
| Jacob
+-----+
6 rows

```

**SELECT \* FROM meus\_contatos**

**WHERE primeiro\_nome = 'Anne' ;**

WHERE diz ao seu software que você quer procurar por algo específico.

Digite isto com seu WHERE e ele dirá que quer procurar apenas valores inseridos na coluna `primeiro_nome`.

Este é o nome da tabela.

= quer dizer é na língua SQL.

Adicione o ponto-e-vírgula e aperte Enter para juntar tudo e perguntar "Se o valor da coluna `primeiro_nome` é Anne, mostre-me o registro".

O valor para sua coluna `primeiro_nome`. Não se esqueça de colocar aspas simples para linhas de textos.

O sistema abaixo mostrará as linhas que foram retornadas pela consulta onde o primeiro nome for Anne.

File Edit Window Help NoDate						
> SELECT * FROM meus_contatos WHERE primeiro_nome = 'Anne';						
sobrenome	primeiro_nome	email	sexo	aniversario	local	
Toth	Anne	Anne Toth@leapinlimos.com	F	NULL	San Fran, CA	
Manson	Anne	am86@objectville.net	F	NULL	Seattle, WA	
Hardy	Anne	anneh@b0tt0msup.com	F	NULL	San Fran, CA	
Parker	Anne	annep@starbuzzcoffee.com	F	NULL	San Fran, CA	
Blunt	Anne	anneblunt@breakneckpizza.com	F	NULL	San Fran, CA	
Jacobs	Anne	anne99@objectville.net	F	NULL	San Jose, CA	

6 rows in set (3.67 sec)



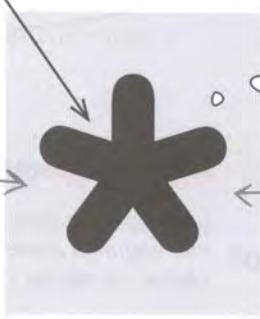
Que \* é esta?

A estrela está dizendo ao sistema SQL para retornar os valores de **todas** as colunas de sua tabela.

**SELECT \* FROM meus\_contatos**

**WHERE primeiro\_nome = 'Anne' ;**

Ao ver o `SELECT *`, pense nele como um pedido ao sistema SQL para SELEÇÃO~~N~~AR TODAS AS COLUNAS.



Eu sou uma estrela!

use a estrela para selecionar todas as colunas da sua tabela.

## não existem Perguntas Idiotas

**P:** E se eu não quiser selecionar todas as colunas? Posso usar outra coisa ao invés do asterisco?

**R:** Certamente que sim. O asterisco seleciona tudo, mas daqui a poucas páginas você aprenderá como selecionar apenas algumas das colunas, fazendo seus resultados ainda mais fáceis de serem interpretados.

**P:** Essa estrela é a mesma coisa que um asterisco?

**R:** Sim, é o mesmo caractere no seu teclado, localizada acima do número 8. Aperte Shift ao mesmo tempo em que o 8 para digitar uma. É o mesmo processo para usuários de Mac e PC.

Embora seja o mesmo caractere que o asterisco, na gíria SQL ela é sempre mencionada como uma estrela. Isto é bom porque dizer "SELECT asterisco from..." não é tão fácil quanto "SELECT estrela from..."

**P:** Há outros caracteres que tenham significado especial como a estrela?

**R:** SQL tem outros caracteres especiais reservados. Você verá mais deste tipo posteriormente neste livro. Mas a estrela é a única que precisa por enquanto. É a única usada na parte juntamente com o comando SELECT no SQL.



**Exercícios**

A recepção do bar Use a Cabeça está adicionando batidas de frutas no menu. Usando o que aprendeu no capítulo 1, escreva os comandos SQL e crie a tabela desta página e em seguida, insira os dados mostrados.

A tabela é parte de um banco de dados chamado **drinks**. Ela contém a tabela **drinks\_faceis** com receitas para um número de bebidas que tem apenas dois ingredientes.

**drinks\_faceis**

nome_do_drink	principal	quantidade1	segundo	quantidade2	instrucoes
Blackthorn	água tônica	1.5	suco de abacaxi	1	mexa com gelo, coloque em uma taça de coquetel com limão batido
Blue moon	soda	1.5	suco de mirtilo	0.75	mexa com gelo, coloque em uma taça de coquetel com limão batido
Oh My Gosh	néctar de pêssego	1	suco de abacaxi	1	mexa com gelo, coloque em um copinho licor
Lime Fizz	Sprite	1.5	suco de limão	0.75	mexa com gelo, coloque em uma taça de coquetel
Kiss on the Lips	suco de cereja	2	néctar de damasco	7	sirva com gelo e canudo
Hot Gold	néctar de pêssego	3	suco de laranja	6	coloque suco de laranja quente em uma caneca e adicione néctar de pêssego
Lone Tree	soda	1.5	suco de cereja	0.75	mexa com gelo, coloque em uma taça de coquetel
Greyhound	soda	1.5	suco de toranja	5	sirva com gelo, mexa bem
Indian Summer	suco de maçã	2	chá quente	6	adicione o suco em uma caneca e completar o resto com chá quente
Bull Frog	chá gelado	1.5	limonada	5	sirva com gelo e uma fatia de limão
Soda and it	soda	2	suco de uva	1	misture em uma taça de coquetel, sem gelo

quantidade1 e quantidade2 estão em onças

→ Resposta na página



**Antes de começar, faça algum planejamento.**

*Escolha seus tipos de dados cuidadosamente e não esqueça sobre o NULL. Então cheque seu código na página 117.*

**Veja Isto!**



Aponte seu lápis

*Não se preocupe sobre as perguntas que ainda não tenha visto. Apenas digite-as como vê, e veja se elas funcionam.*

## DÊ UM NOME AO DRINK

Use a tabela `drinks_faceis` que acabou de criar e tente usar estas consultas na sua máquina. Escreva quais drinks ele retornou como resposta de cada consulta.



```
SELECT * FROM drinks_faceis WHERE main = 'Sprite';
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE main = soda;
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE amount2 = 6;
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE second = "suco de laranja";
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE amount1 = 1.5;
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE amount2 < '1';
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE main = 'soda' ;
```

Que drink(s)? .....

```
SELECT * FROM drinks_faceis WHERE amount1 < '1.5' ;
```

Que drink(s)? .....

Para pontos extras, escreva aqui qual consulta não funciona...

**Sim, você está exatamente certa.**

Uma destas consultas não funciona, as outras funcionam, mas o resultado delas poderá não ser o que espera.

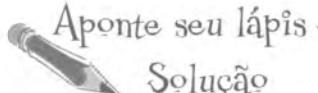
Espere um segundo... "Tente estas consultas", você disse. Fez parecer que todas iriam funcionar. E eu acreditei! Mas uma delas não funciona. E algumas delas não parecem que vão funcionar.



... e quais funcionaram e que você esperava que não fossem funcionar.

gnificado  
eciais ou  
eriormente  
cisa saber  
junto com

de  
de  
o de  
de  
na  
de  
jelo



Aponte seu lápis

Solução

# DÊ UM NOME AO DRINK



Você testou estas consultas na sua tabela `drinks_faceis` e anotou quais drinks retornaram como resposta para cada consulta.

`SELECT * FROM drinks_faceis WHERE main = 'Sprite';`

Que drink(s)? Lime Fizz



Note o uso das aspas simples

`SELECT * FROM drinks_faceis WHERE main = soda;`

Que drink(s)? Error Hum, parece que esta é a consulta que não queria funcionar.

`SELECT * FROM drinks_faceis WHERE amount2 = 6;`

Que drink(s)? Hot Gold, Indian Summer

← Esta aqui funciona, ela é uma variável DEC e por isso não precisa usar as aspas.

`SELECT * FROM drinks_faceis WHERE second = "suco de laranja";`

Que drink(s)? Hot Gold, Grey Hound

`SELECT * FROM drinks_faceis WHERE amount1 = 1.5;`

Que drink(s)? Blackthon, Bluemoon, Lime Fizz, Lone Tree, Grey Hound, Bull Frog

`SELECT * FROM drinks_faceis WHERE amount2 < '1';`

Que drink(s)? Blue Moon, Lime Fizz, Lone Tree

SELECT \* FROM drinks\_faceis WHERE main = 'soda';

Outra forma correta  
da cláusula WHERE

Que drink(s)? Blue Moon, Lone Tree, Grey Hound, Soda and it

`SELECT * FROM drinks_faceis WHERE amount1 < '1.5';`

Que drink(s)? Oh My Gosh

Para pontos extras, escreva aqui qual consulta não funciona...

→ **WHERE main = soda;**

Esta é a cláusula WHERE que não funciona. Você precisa de aspas no objeto de pesquisa, por se tratar do tipo VARCHAR.

... e quais funcionaram e que você esperava que não fossem funcionar.

→ **WHERE second = "suco de laranja";**

Esta consulta funciona e não causa erro, mas retorna a consulta sem resultados. SQL espera que use aspas simples, como fez ao inserir o valor.

→ **WHERE amount2 < '1';**

Esta aqui funciona, ainda que não devesse, pois as variáveis DEC não precisam do uso de aspas.

→ **WHERE amount1 = '1.5';**

Este aqui também funciona!

Estas duas últimas pesquisas funcionarão porque a maioria dos Sistemas SQL dá um pouco de amplitude. Eles ignoram as aspas e tratam os valores tipo DEC e INT como números, ainda que as aspas indiquem que são valores de texto. As consultas NÃO ESTÃO CORRETAS, mas seu Sistema SQL está perdoando.

## Como fazer consultas nos seus tipos de dados

Para escrever cláusulas WHERE válidas, você precisa certificar-se que cada tipo de dados que incluir está formatado corretamente. Aqui estão as convenções para os principais tipos de dados:



Nós ❤️ aspas simples	Sem aspas para a gente
CHAR	DEC
VARCHAR	INT
DATE	
DATETIME, TIME, or TIMESTAMP	
BLOB	

## Mais problemas de pontuação

Greg conseguiu mais alguns contatos ontem à noite. Ele está tentando inseri-los em sua tabela:

```
INSERT INTO meus_contatos
```

```
VALUES
```

```
('Funyon', 'Steve', 'steve@onionflavoredrings.com', 'M',
'01-04-1970', 'Punk', 'Grover's Mill, NJ', 'Solteiro',
'esmagar o estado', 'compatriotas, guitarristas');
```

STEVE FUNYON  
Aniversário: 01/04/1970  
Punk  
Solteiro  
Grover's Mill, NJ  
steve@onionflavoredrings.com  
Interesses: esmagar o estado  
Procura: compatriotas, guitarristas

Mas o programa de parecer não estar funcionando. Ele digita alguns ponto e vírgulas para fazer o Sistema SQL finalizar. Falta de sorte.

```
File Edit Window Help Aliens!
> INSERT INTO meus_contatos VALUES ('Funyon', 'Steve', 'steve@onionflavoredrings.com', 'M', '01-04-1970', 'Punk', 'Grover's Mill, NJ', 'Solteiro', 'esmagar o estado', 'compatriotas, guitarristas');
'>
'>;
'>;
'>
```

*Ele vê este prompt toda vez que aperta Enter:>*



### PODER DO CÉREBRO

O que acha que está acontecendo aqui?

Hum, olha aquelas aspas simples que ficam aparecendo antes do prompt. Aposto que há algo de errado com as aspas em nosso comando INSERT...



## Aspas simples incompatíveis

Exatamente! Quando Greg tentou inserir o registro, o Sistema SQL estava esperando um número par de aspas simples, uma antes e uma depois de cada valor VARCHAR, CHAR e DATE. O nome da cidade, **Grover's Mill**, confundiu o assunto porque foi adicionado um apóstrofo extra. O sistema SQL ainda está esperando por uma aspa simples para o fechamento do texto.



### Você pode voltar ao controle de seu sistema.

Finalize o comando digitando uma aspa simples e um ponto e vírgula. Isto dará ao Sistema SQL a aspa simples a mais que ele está esperando.

Você terá uma mensagem de erro quando digitar a outra aspa e o ponto-e-vírgula e terá que inserir o comando INSERT de novo a partir do rascunho.

*Você obterá um erro →  
apos fazer isto, mas  
pelo menos poderá  
tentar de novo.*

Ao inserir a aspa simples e um ponto e-virgula o comando **INSERT** interrompido será finalizado.

Este erro dá uma ideia bem clara sobre o que está errado. Ele cita parte da sua consulta iniciando com a aspa simples a mais.

Mesmo que o registro não tenha sido inserido, o último > mostra que pelo menos o programa está respondendo novamente.

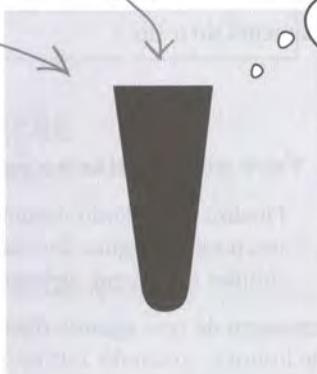
**Aspas simples são caracteres especiais**

Ao tentar inserir um VARCHAR, CHAR ou BLOB que contenha um apóstrofo, você deve indicar ao seu Sistema SQL que ele não significa o final daquele texto, **mas sim parte do texto** e precisa ser incluído na linha. Uma maneira de fazer isso é **adicionando uma barra invertida** na frente da aspa simples.

```
INSERT INTO meus_contatos  
(local)  
VALUES ('Grover\'s Mill');  
( 'Grover\'s Mill');
```

A aspa simples é um caractere especial reservado. Isto significa que ela tem uma função especial na linguagem.

Ela é usada para dizer ao seu Sistema SQL onde uma linha de texto começa e termina.



Eu sou a aspas simples  
e quando estou sozinha,  
eu preciso de uma barra  
invertida.

---

não existem  
Perguntas Idiotas

---

P: Isto não é a mesma coisa que um apóstrofo?

R: Exatamente a mesma coisa que um apóstrofo; no entanto, SQL designa um significado muito especial a ele. É usado ao software SQL que os dados entre dois deles são os dados do texto.

P: Que tipos de dados precisam dele?

R: Os tipos de dados de texto. Os dados de texto simplesmente significam que os dados são de uma coluna VARCHAR, CHAR, BLOB ou TIMEDATE. Qualquer um que não seja um número.

P: As colunas DEC e INT precisam dele?

R: Não. As colunas numéricas não precisam de espaços, então é fácil dizer quando os números acabam e a próxima palavra do código começa.

P: Então, ela só é usada para colunas de texto?

R: Sim. O único problema é que colunas de texto têm espaços, o que causa problemas quando seus dados contêm apóstrofos. SQL não sabe como distinguir entre o apóstrofo dentro da coluna e apóstrofo que diz ao Sistema que a coluna começou ou acabou.

P: Não poderíamos facilitar dizendo separadamente ao usar aspas duplas ao invés de aspas simples?

R: Não. Não use aspas duplas no caso de, posteriormente, usar seus comandos SQL combinados com uma linguagem de programação (como PHP). Você usa "na linguagem de programação para dizer "aqui é onde o comando SQL fica"; desta forma, aspas simples são reconhecidas como sendo parte daquele comando e não parte da linguagem de programação.

## INSERT (Insira) dados com aspas simples nele

Você precisa dizer ao seu Sistema SQL que aquela aspa simples não está ali para iniciar ou terminar uma linha de texto, mas sim que é *parte* da linha de texto.

### Manuseie as aspas com a barra invertida

Você pode fazer isto (e consertar seu comando INSERT ao mesmo tempo) ao adicionar o caractere barra invertida na frente da aspa simples integrante na sua linha de texto:

```
INSERT INTO meus_contatos
VALUES
```

```
('Funyon', 'Steve', 'steve@onionflavoredrings.com', 'M',
'01-04-1970', 'Punk', 'Grover\'s Mill, NJ', 'Solteiro',
'esmagar o estado', 'compatriotas, guitarristas');
```

*Dizer ao Sistema SQL que uma aspa simples é parte do texto ao colocar uma barra invertida na frente dela é chamado de "escapar."*

### Manuseie as aspas com uma aspa simples extra

Outra maneira de "escapar" aspas é colocar uma aspa simples extra na frente dela.

```
INSERT INTO meus_contatos
VALUES
```

```
('Funyon', 'Steve', 'steve@onionflavoredrings.com', 'M',
'1970-04-01', 'Punk', 'Grover''s Mill, NJ', 'Solteiro',
'esmagar o estado', 'compatriotas, guitarristas');
```

*Ou você pode "escapar" uma aspa simples colocando uma aspa extra na frente dela.*



Que outros caracteres podem causar o mesmo problema?



Se você tem dados na sua tabela com aspas, pode ser que tenha que procurar por eles com a cláusula WHERE em algum momento. Para usar o SELECT para dados contendo aspas simples com sua cláusula WHERE, você precisa "escapar" sua aspa simples da mesma forma que fez ao inseri-la. Reescreva o código abaixo usando métodos diferentes para "escapar" a aspas simples.

```
SELECT * FROM meus_contatos  
WHERE  
location = 'Grover's Mill, NJ';
```

**1**

.....  
.....  
.....

**2**

.....  
.....  
.....

**Que método você prefere?**



**Solução dos  
Exercícios**

Se você tem dados na sua tabela com aspas, pode ser que tenha que procurar por eles com a cláusula WHERE em algum momento. Para usar o SELECT para dados contendo aspas simples com sua cláusula WHERE, você precisa "escapar" sua aspa simples da mesma forma que fez ao inseri-la. Reescreva o código abaixo usando métodos diferentes para escapar a aspas simples.

```
SELECT * FROM meus_contatos  
WHERE  
location = 'Grover's Mill, NJ';
```

**1**

**SELECT\* FROM meus\_contatos  
WHERE  
local = 'Grover\'s Mill, NJ';**

Método 1, a barra invertida

**2**

**SELECT\* FROM meus\_contatos  
WHERE  
local = "Grover... "S... Mill, NJ";**

Método 2, a aspa simples extra.

## Selecionar (**SELECT**) dados específicos

Agora que dominou como selecionar todos os tipos de dados com aspas, e como usar o **SELECT** em dados onde há aspas.

Espere. Toda vez que uso um comando **SELECT \* meus\_contatos** é uma grande confusão porque ele se enrola. Posso esconder todas aquelas colunas extras quando, talvez, eu queira apenas o email de alguém?



**Você precisa saber usar o **SELECT** (selecionar) apenas com as colunas que deseja ver.**

O que precisamos é maior precisão. Vamos tentar limitar os resultados um pouco. Limitar os resultados significa obter menos colunas no nosso resultado. Nós selecionamos apenas as colunas que desejamos ver.



Exercícios

### Tente em casa

Antes de testar esta consulta **SELECT**, rascunhe como acha que a tabela com os resultados será (se precisar dar uma olhada na tabela **drinks\_faceis**, você pode encontrá-la na página 59.)

Nós substituímos o \* pelo nome destas colunas.

```
SELECT nome_do_drink, principal, segundo
FROM drinks_faceis
WHERE principal = 'soda';
```



Solução dos Exercícios

### Tente em casa

Antes de testar esta consulta **SELECT**, rascunhe como acha que a tabela com os resultados será.

nome_do_drink	principal	segundo
Blue moon	soda	suco de mirtilo
Lone Tree	soda	suco de cereja
Greyhound	soda	suco de toranja
Soda and it	soda	suco de uva

## O jeito antigo

```
SELECT * FROM drinks_faceis;
```

Aqui obtivemos todas as colunas e o nosso resultado é grande demais para caber na janela do nosso sistema. Eles se enrolam para a próxima linha e o resultado é uma bagunça.

nome_do_drink	principal	quantidade1	segundo	quantidade2	instrucoes
Kiss on the Lips	suco de cereja	2.0	néctar de damasco	7.00	Sirva com gelo e canudo
Hot Gold	néctar de pêssego	3.0	suco de laranja	6.00	coloque suco de laranja quente numa caneca e adicione néctar de pêssego
Lone Tree	soda	1.5	suco de cereja	0.75	Mexa com gelo, coloque em uma taça de coquetel
Greyhound	soda	1.5	suco de toranja	5.00	Sirva com gelo, mexa bem
Indian Summer	suco de maçã	2.0	chá quente	6.00	adiciona suco a uma caneca e complete o resto com chá quente
Bull Frog	chá gelado	1.5	limonada	5.00	Sirva com gelo e uma fatia de lima
Soda and It	soda	2.0	suco de uva	1.00	misture em uma taça de coquetel, sem gelo
Blackthorn	Água tônica	1.5	Suco de abacaxi	1.00	Mexa com gelo, coloque em uma taça de coquetel com limão batido
Blue Moon	soda	1.5	suco de mirtilo	0.75	Mexa com gelo, coloque em uma taça de coquetel com limão batido
Oh My Gosh	néctar de pêssego	1.0	suco de abacaxi	1.00	Mexa com gelo, coloque em um copinho de licor
Lime Fizz	Sprite	1.5	suco de limão	0.75	Mexa com gelo, coloque em uma taça de coquetel

11 rows in set (0.00 sec)

Sele  
Ao especi  
valores da  
de linhas,  
deixar o S

SE  
FF

Sele  
Esta é um  
À medida  
resultado  
SQL con

## Selecionar (**SELECT**) colunas específicas para limitar os resultados

Ao especificar quais colunas queremos que retornem na nossa consulta, podemos escolher apenas os valores das colunas que precisamos. Assim como utiliza uma cláusula WHERE para limitar o número de linhas, você pode usar uma seleção de coluna para limitar o número de colunas. A questão é deixar o Sistema SQL fazer o trabalho pesado por você.

```
SELECT nome_do_drink, principal, segundo
FROM drinks_faceis;
```

...mas podemos limitar nossos resultados ao selecionar apenas as colunas que queremos exibir no resultado.

```

File Edit Window Help JustEnough
> SELECT nome_do_drink, principal, segundo FROM drinks_faceis;
+-----+-----+-----+
| nome_do_drink | principal | segundo |
+-----+-----+-----+
| Kiss on the Lips | suco de cereja | néctar de damasco |
| Hot Gold | néctar de pêssego | suco de laranja |
| Lone Tree | soda | suco de cereja |
| Greyhound | soda | suco de toranja |
| Indian Summer | suco de maçã | chá quente |
| Bull Frog | chá gelado | limonada |
| Soda and It | soda | suco de uva |
| Blackthorn | água tônica | Suco de abacaxi |
| Blue Moon | soda | suco de mirtilo |
| Oh My Gosh | néctar de pêssego | Suco de abacaxi |
| Lime Fizz | Sprite | suco de limão |
+-----+-----+-----+
11 rows in set (0.00 sec)

```

## Selecione (**SELECT**) colunas específicas para resultados mais rápidos

Esta é uma boa prática de programação a se seguir, mas tem outros benefícios. À medida que suas tabelas ficam maiores, ela acelera a recuperação de seus resultados. Você verá também mais velocidade quando, eventualmente, usar SQL com outra linguagem de programação, como a PHP.



Aponte seu lápis \_\_\_\_\_



Termine

## Muitas maneiras de conseguir um drink Kiss on The Lips

Ainda lembra de nossa tabela `drinks_faceis`? Este comando SELECT resultará em um Kiss on the Lips:

```
SELECT nome_do_drinks FROM drinks_faceis
```

**WHERE**

**Principal = 'suco de cereja'**

Termine os outros quatro códigos na próxima página para conseguir um Beijo também.

`drinks_faceis`

nome_do_drink	principal	quantidade1	segundo	quantidade2	instruções
Blackthorn	água tônica	1.5	suco de abacaxi	1	mexa com gelo, coloque em uma taça de co... com limão batido
Blue moon	soda	1.5	suco de mirtilo	0.75	mexa com gelo, coloque em uma taça de co... com limão batido
Oh My Gosh	néctar de pêssego	1	suco de abacaxi	1	mexa com gelo, coloque em um copinho de...
Lime Fizz	Sprite	1.5	suco de limão	0.75	mexa com gelo, coloque em uma taça de co...
Kiss on the Lips	suco de cereja	2	néctar de damasco	7	Sirva com gelo e canudo
Hot Gold	néctar de pêssego	3	suco de laranja	6	coloque suco de laranja quente em uma can... adicone néctar de pêssego
Lone Tree	soda	1.5	suco de cereja	0.75	mexa com gelo, coloque em uma taça de co...
Greyhound	soda	1.5	suco de toranja	5	sirva com gelo, mexa bem
Indian Summer	suco de maçã	2	chá quente	6	adicionar o suco em uma caneca e comple... resto com chá quente
Bull Frog	chá gelado	1.5	limonada	5	sirva com gelo e uma fatia de limão
Soda and it	soda	2	suco de uva	1	misture em uma taça de coquetel, sem gelo

**SELECT** .....

**WHERE** .....

**SELECT** .....

**WHERE** .....

**SELECT** .....

**WHERE** .....

**SELECT** .....

**WHERE** .....

Agora escreva os três comandos SELECT que darão a você um BULL FROG.

1

.....

2

.....

3

.....

- Use asp... for seleci...
- Não use númer... seleciona...

P: E se eu preciso das colunas de...  
levaria nomeá-las?

R: Se precisar de... quando não precis... isá-la.

P: Tentei copiar e... iquei recebendo er... azendo algo errado

R: Consultas co... ezes contêm car... spaços, mas sig...



## Aponte seu lápis Solução

Termine os outros quatro códigos na próxima página para conseguir um Kiss também.

`SELECT nome do drink FROM drinks faceis`

`WHERE Segundo = 'néctar de damasco';`

`SELECT nome do drink FROM drinks faceis`

`WHERE quantidadeZ = 7;`

`SELECT nome do drink FROM drinks faceis`

`WHERE instrucoes = 'sirva com gelo e canudo';`

`SELECT nome do drink FROM drinks faceis`

`WHERE nome do drink = 'Kiss on the Lips';`

↑  
Isso é aquele que raramente vai usar, mas que vai dar o resultado que você quer. Dever-se usar algo parecido com isso quando quiser ter certeza que sua coluna nome do drink não terá erro de digitação.

Agora escreva os três comandos SELECT que darão a você um BULL FROG.

1 `SELECT nome do drink FROM drinks faceis`

`WHERE principal = 'chá gelado';`

2 `SELECT nome do drink FROM drinks faceis`

`WHERE segundo = 'limonada';`

3 `SELECT nome do drink FROM drinks faceis`

`WHERE instrucoes = 'sirva com gelo e uma fatia de lima';`

## PONTOS DE BALA



- Use aspas simples nas suas cláusulas WHERE quando for selecionar campos de textos.
- Não use aspas simples quando for selecionar campos numéricos.
- Use o \* no seu comando SELECT quando quiser selecionar todas as colunas.

- Se tiver inserido sua consulta e o Sistema SQL não finalizar o processo, procure se há falta de uma aspa simples.
- Quando puder, selecione colunas específicas na sua tabela ao invés de usar SELECT \*.

## não existem Perguntas Idiotas

P: E se eu precisar que a consulta retorne todas as colunas da minha tabela? Eu realmente deveria nomeá-las ao invés de usar o \*?

R: Se precisar de todas elas, então use \*. Apenas quando não precisar de todas elas que não deve usá-la.

P: Tentei copiar e colar uma consulta da Internet e fiquei recebendo erros quando tentava usá-la. Estou fazendo algo errado?

R: Consultas copiadas de navegadores web às vezes contêm caracteres invisíveis que parecem espaços, mas significam algo a mais para a SQL.

Colá-las em um editor de texto é uma forma de ver e remover estes caracteres "gremlins". Sua melhor alternativa é colar tudo em um editor de texto e analisar cuidadosamente.

P: Então eu deveria colar em algum lugar como Microsoft Word, por exemplo?

R: Não, Word não é uma boa opção, uma vez que ele não faz nada para exibir as formatações invisíveis que podem estar no texto. Tente o Bloco de Notas (PC) ouTextEdit no modo texto sem formatação (MAC).

P: Sobre escapar o apóstrofo, há alguma razão em usar um método ao invés de outro?

R: Na verdade não. Nós tendemos a usar o método da barra invertida somente porque achamos que é mais fácil para visualizar onde o apóstrofo extra está quando alguma coisa der errado na consulta. Por exemplo, isto é mais fácil de processar visualmente:

'Isn\'t that your sister\'s pencil?'  
Do que isto:

'Isn\'t that your sister''s pencil?'

Além disso, não há razão alguma em beneficiar um método em detrimento a outro. Ambos os métodos permitem que você insira um apóstrofo nas suas colunas de texto.

# Donuts pergunta o que sua tabela pode fazer por você...

Para encontrar o melhor donut com cobertura na tabela, você precisa fazer pelos menos dois comandos SELECT. O primeiro selecionará linhas com o tipo correto de donut. A segunda vai selecionar a linha com donuts com a nota 10.

Eu quero encontrar o melhor donut com cobertura da cidade, sem precisar ficar caçando todos os resultados.



**notas\_donuts**

empresa	hora	data	tipo	nota	comentários
Starbuzz Coffee	07:43	23/04	cobertura de canela	6	muito condimentado
Duncan's Donuts	08:56	25/04	cobertura simples	5	gorduroso
Duncan's Donuts	19:58	24/04	geléia	6	não são fresquinhos mas são gostosos
Starbuzz Coffee	22:35	23/04	cobertura simples	7	morno, mas não é quente
Krispy King	21:39	26/09	geléia	6	sem geléia suficiente
Starbuzz Coffee	7:48	23/04	chocolate com castanhas	10	marshmallows
Krispy King	8:56	25/11	cobertura simples	8	cobertura de xarope

Imagine que esta tabela tenha 10.000 registros.

- 1 Uma maneira é procurar pelo tipo de donut:

`SELECT empresa, nota FROM notas_donuts  
WHERE`

`Tipo = 'cobertura simples';`

Todos os resultados serão do tipo correto de donut

Você precisa SELECT (selecionar) a nota para procurar dentro as notas mais altas e empresas para conseguir o nome da vencedora.

empresa
Duncan's Donut
Starbuzz Coffee
Krispy King
Starbuzz Coffee
Duncan's Donut

Aqui está o resultado da primeira consulta, mas imagine centenas de resultados.

## Pergunte o que você pode fazer por seu donut

- 1 Ou então você precisa procurar pelas notas mais altas:

`SELECT empresa, tipo FROM notas_donuts`

`WHERE`

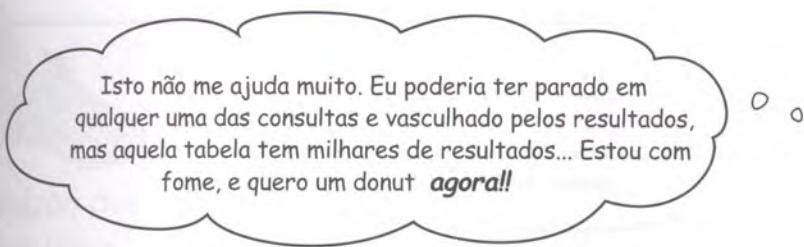
`Nota = 10;`

Todos os resultados serão as notas máximas.

Você precisa procurar por todos os tipos e então a empresa lhe dará o nome do vencedor.

empresa	tipo
Starbuzz Coffee	chocolate com castanhas
Krispy King	cobertura simples
Starbuzz Coffee	cobertura simples
Duncan's Donuts	chocolate com castanhas

O resultado da segunda consulta, novamente, retrata centenas de resultados.



### PODER DO CÉREBRO

Em português claro, que pergunta você realmente quer fazer para essas consultas?

## Combinando suas consultas

Podemos dar conta das duas coisas que estamos procurando, ‘cobertura simples’, como tipo, e 10, como nota, em uma só consulta usando a palavra-chave AND (e). Os resultados que obtivermos satisfarão ambas as condições.

**SELECT empresa**

Agora tudo o que precisamos é  
SELECT a empresa

**FROM notas\_donuts**

**WHERE tipo = 'cobertura simples'**

**AND**

Use a palavra AND para combinar suas duas cláusulas WHERE

**Nota= 10;**

←

Aqui está o resultado da consulta AND. Mesmo se recebermos mais de uma linha com resultado da pesquisa, saberíamos que todas as empresas do resultado têm donuts de cobertura simples com a nota 10, então você poderia ir a qualquer uma delas. Ou a todas elas.

empresa	nota
Duncan's Donut	5
Starbuzz Coffee	7
Krispy King	8
Starbuzz Coffee	10
Duncan's Donut	8

**AND**

empresa	tipo
Starbuzz Coffee	chocolate com castanhas
Krispy King	cobertura simples
Starbuzz Coffee	cobertura simples
Duncan's Donuts	chocolate com castanhas

Esta consulta combina os resultados de 'cobertura simples e nota = 10' para encontrar qualquer resultado que se adapte a ambas as consultas.

Mamãe?  
Podemos ir ao Starbuzz? Por favooooor!?

empresa
Starbuzz Coffee





Então eu poderia ter encontrado Anne  
usando AND?

Utilizando a tabela `meus_contatos`, escreva algumas consultas para Greg. `SELECT` (selecione) apenas as colunas que realmente precisar para obter sua resposta. Preste atenção nas aspas simples.

Escreva uma consulta para encontrar o email de todos os programadores.

.....  
.....  
.....  
.....



Escreva uma consulta para encontrar o nome e local de qualquer um com seu aniversário (data do nascimento).

.....  
.....  
.....  
.....

Escreva uma consulta para encontrar o nome e email de qualquer pessoa solteira na sua cidade. Para pontos extras, escolha apenas aquelas do sexo que você gostaria de ter um encontro.

.....  
.....  
.....  
.....

Escreva uma consulta que Greg poderia ter utilizado para encontrar todas as Annes de São Francisco.

.....  
.....  
.....  
.....



## Solução dos Exercícios

Utilizando a tabela meus\_contatos, escreva algumas consultas para Greg. SELECT (selecione) apenas as colunas que realmente precisar para obter sua resposta. Preste atenção nas aspas simples.

Escreva uma consulta para encontrar o email de todos os programadores.

Nos queremos a coluna e-mail. `SELECT email FROM meus_contatos WHERE profissão = 'programador';`

A profissão que nós queremos é programador.

Escreva uma consulta para encontrar o nome e local de qualquer um com seu aniversário (data do nascimento).

`SELECT sobrenome, primeiro_nome, local  
FROM meus_contatos  
WHERE birthday = '05-09-1975';`

Aqui deveria ser seu aniversário entre aspas.

Escreva uma consulta para encontrar o nome e email de qualquer pessoa solteira na sua cidade. Para pontos extras, escolha apenas aquelas do sexo que você gostaria de ter um encontro.

`SELECT sobrenome, primeiro_nome, email  
FROM meus_contatos  
WHERE local = 'San Antonio, TX'  
AND sexo = 'M';`

Sua cidade aqui.  
O sexo da pessoa que você gostaria de ter um encontro.

Escreva uma consulta que Greg poderia ter utilizado para encontrar todas as Annes de São Francisco.

`SELECT sobrenome, primeiro_nome, email  
FROM meus_contatos  
WHERE local = 'San Fran, CA'  
AND primeiro_nome = 'Anne';`

Olhando para trás na tabela, Greg parece ter encurtado São Francisco para San Fran. Esperamos que ele tenha sido consistente.

## Encontrando valores numéricos

Digamos que queremos encontrar todos os drinks da tabela `drinks_faceis` que contenham mais de uma onça em uma única consulta. Aqui está o jeito difícil para encontrar os resultados. Você pode utilizar duas consultas:

*Apenas queremos o nome dos drinks.*

```
SELECT nome_do_drink FROM drinks_faceis
WHERE
Principal = 'soda'
AND
quantidade1 = 1.5;
```

*Drinks com soda com 1.5 onça de soda.*

```
File Edit Window Help MoreSoda
> SELECT nome_do_drink FROM drinks_faceis WHERE Principal =
'soda' AND quantidade1 = 1.5;
+-----+
| nome_do_drink |
+-----+
| Blue Moon     |
| Lone Tree    |
| Greyhound    |
+-----+
3 rows in set (0.00 sec)
```

*Drinks de soda com 2 onças de soda.*

```
SELECT nome_do_drink FROM drinks_faceis
WHERE
Principal = 'soda'
AND
quantidade1 = 2;
```

```
File Edit Window Help EvenMoreSoda
> SELECT nome_do_drink FROM drinks_faceis WHERE Principal =
'soda' AND quantidade1 = 2;
+-----+
| nome_do_drink |
+-----+
| Soda and It   |
+-----+
1 row in set (0.00 sec)
```



Não seria um sonho se eu só encontrasse todos os drinks na tabela drinks\_faceis que contenham mais do que uma onça em uma só consulta. Mas sei que isto é só uma fantasia...

drinks\_faceis

nome_do_drink	principal	quantidade1	segundo	quantidade2	instrucoes
Blackthorn	água tônica	1.5	suco de abacaxi	1	mexa com gelo, coloque em uma taça de coquetel com limão batido
Blue moon	soda	1.5	suco de mirtilo	0.75	mexa com gelo, coloque em uma taça de coquetel com limão batido
Oh My Gosh	néctar de pêssego	1	suco de abacaxi	1	mexa com gelo, coloque em um copinho de licor
Lime Fizz	Sprite	1.5	suco de limão	0.75	mexa com gelo, coloque em uma taça de coquetel
Kiss on the Lips	suco de cereja	2	néctar de damasco	7	sirva com gelo e canudo
Hot Gold	néctar de pêssego	3	suco de laranja	6	coloque suco de laranja quente em uma caneca e adicione néctar de pêssego
Lone Tree	soda	1.5	suco de cereja	0.75	mexa com gelo, coloque em uma taça de coquetel
Greyhound	soda	1.5	suco de toranja	5	sirva com gelo, mexa bem
Indian Summer	suco de maçã	2	chá quente	6	adicionar o suco em uma caneca e completar o resto com chá quente
Bull Frog	chá gelado	1.5	limonada	5	sirva com gelo e uma fatia de lima
Soda and it	soda	2	suco de uva	1	misture em uma taça de coquetel, sem gelo

## Uma vez é o suficiente

É uma perda de tempo usar duas colunas, e ainda poderá deixar passar bebidas com quantidades como 1.75 ou 3 onças. Em vez disso, pode-se usar um sinal **de maior**:

**SELECT nome\_do\_drink FROM drinks\_faceis**

**WHERE**

**principal = 'soda'**

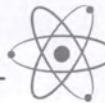
**AND**

**quantidade1 > 1;**

O sinal DE MAIOR lhe dará todos os drinks que tenham mais do que / onça de soda.



```
File Edit Window Help DoltOnce
SELECT nome_do_drink FROM drinks_faceis WHERE principal =
'soda' AND quantidade1 > 1;
+-----+
| nome_do_drink |
+-----+
| Blue Moon     |
| Lone Tree     |
| Greyhound     |
| Soda and It   |
+-----+
4 rows in set (0.00 sec)
```



## PODER DO CÉREBRO

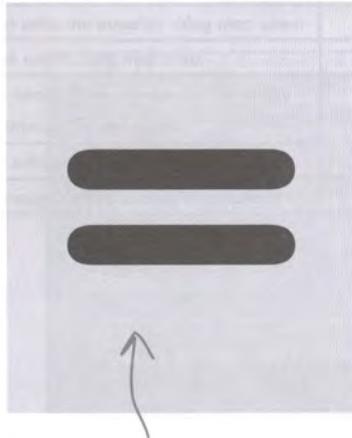
Por que você não pode combinar as duas consultas com um AND adicional?

## Operadores de comparação ~~sem problemas~~

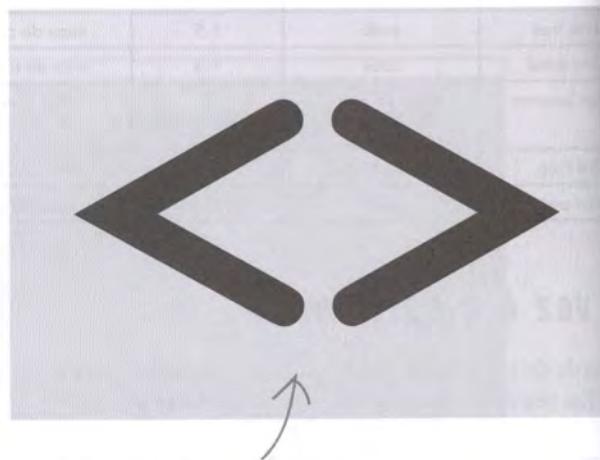
Até agora, você tem usado apenas o sinal de **igual** na sua cláusula WHERE. Você acabou de ver o sinal **de maior**, **>**. O que ele faz é comparar um valor em relação a outro. Aqui estão os outros operadores de comparação.

O sinal de igual procura por compatíveis exatos. Ele não nos serve quando queremos encontrar algo que seja maior ou menor que a outra coisa.

Esse sinal confuso **não é igual (também conhecido como diferente)**. Ele retorna exatamente os resultados opostos que o sinal de igual. Dois valores podem ser tanto iguais ou não iguais.



Todos já conhecemos e amamos o sinal de IGUAL.



Este aqui quer dizer NÃO IGUAL. Ele retorna todos os registros que não condizem com a condição.

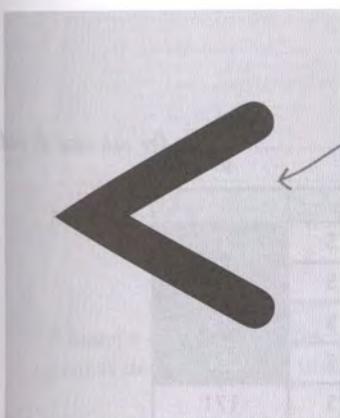


## Musculação cerebral

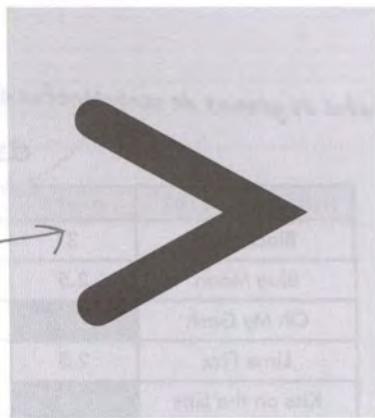
Você já percebeu que toda cláusula WHERE até agora tem sempre um nome de coluna na esquerda. Ela funcionaria se o nome da coluna estivesse na direita?

O **sinal de menor** procura pelos valores na coluna na esquerda e os compara com os valores da direita. Se o valor da coluna é menor que o valor da direita, aquela linha é exibida.

O **sinal de maior** é o inverso do sinal de menor. Ele procura por valores na coluna e os compara aos valores da coluna da direita. Se o valor da coluna é maior que o valor da direita, aquela linha é exibida.



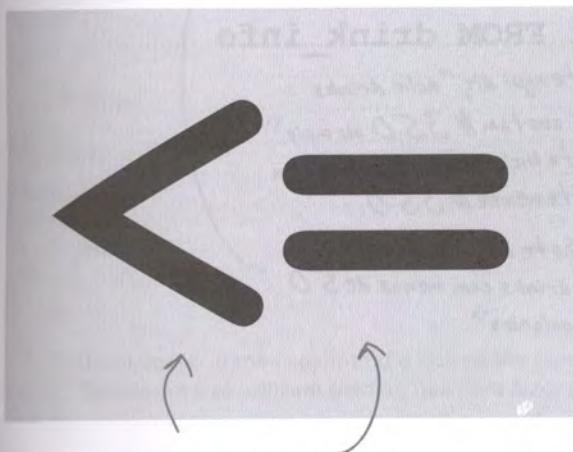
SINAL DE MENOR exibe todos os valores menores que a condição.



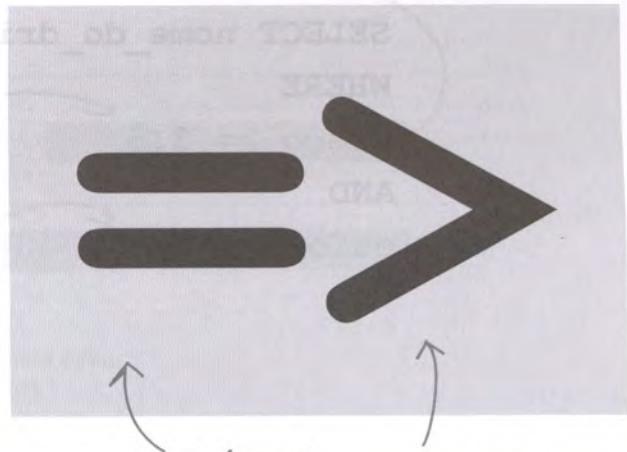
E, é claro, há também o SINAL DE MAIOR.

A única diferença com o **sinal de menor ou igual** é que os valores das colunas iguais a condição também são exibidos.

A mesma coisa com o **sinal de maior ou igual**. Se o valor da coluna for igual ou maior que o valor da condição, a linha será exibida.



Qualquer valor que for MENOR OU IGUAL A a condição é exibida.



E há também o sinal DE MAIOR OU IGUAL.

## Encontrando dados numéricos com operadores de comparação

O bar do Use a Cabeça tem uma tabela com o custo e informação nutricional de seus drinks. Eles querem exibir os drinks mais caros e com menos calorias para aumentarem os lucros.

Eles estão usando operadores de comparação para encontrar os drinks que custam pelo menos \$ 3.50 e têm menos que 50 calorias na tabela `drink_info`.

O total de gramas de carboidratos em cada drink.

`drink_info`

As calorias de cada

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

`SELECT nome_do_drink FROM drink_info`

`WHERE`

`preço >= 3.5`

Este aqui diz "ache drinks que custam \$ 3.50 ou mais."  
Isto inclui drinks que custam exatamente \$ 3.50.

`AND`

`calorias < 50;`

Este diz: "encontre drinks com menos de 50 calorias".

Esta consulta só exibe drinks em que **ambas** condições são encontradas por causa do AND combinando os dois resultados. Os drinks que são exibidos são: Oh My Gosh, Lone Tree e Soda and It.

Aponte seu lápis \_\_\_\_\_

Sua vez de fazer uma mistura. Escreva consultas que retornarão as seguintes informações. Escreva também qual foi o resultado de cada consulta:

O custo de cada drink com gelo que seja amarelo e com mais de 33 calorias.

.....  
.....  
.....  
.....  
.....

Resultado: .....

O nome e cor de cada drink que não contenham mais que 4 gramas de carboidrato e utilizam gelo.

.....  
.....  
.....  
.....  
.....

Resultado: .....

O custo de cada drink que tenha 80 calorias ou mais.

.....  
.....  
.....  
.....  
.....

Resultado: .....

Drinks com o nome Greyhound e Kiss on the Lips, seguido da cor de cada um e se utilizam gelo ou não para fazer a bebida.

.....  
.....  
.....  
.....  
.....

Resultado: .....


**Aponte seu lápis**  
**Solução**

Sua vez de fazer uma mistura. Escreva consultas que retornarão as seguintes informações. Escreva também qual foi o resultado de cada consulta:

O custo de cada drink com gelo que seja amarelo e com mais de 33 calorias.

```
SELECT preco FROM drink_info
WHERE gelo = 'S'
AND
cor = 'amarelo'
AND
Calorias > 33;
```

Resultado: **\$4.00**

O nome e cor de cada drink que não contenham mais que 4 gramas de carboidrato e utilizam gelo.

```
SELECT nome do drink FROM drink_info
WHERE
Carboidratos <= 4
AND
Gelo = 'S';
```

Resultado: **Blue Moon, blue**

O custo de cada drink que tenha 80 calorias ou mais.

```
SELECT preco FROM drink_info
WHERE
Calorias >= 80;
```

Resultado: **\$5.50, \$3.20, \$2.60**

Drinks com o nome Greyhound e Kiss on the Lips, seguido da cor de cada um e se utilizam gelo ou não para fazer a bebida.

```
SELECT nome do drink, cor, gelo FROM drink_info
WHERE
Preco > 3.8;
```

Resultado: **Kiss on the Lips, rosa, S  
Greyhound, amarelo, S  
Soda and It, vermelho, N**

Mas isto só funciona com números correto? Se eu quiser encontrar todos os drinks cujos nomes começam com uma letra específica, eu estaria sem sorte?

Este aqui é uma pegadinha.  
Você tinha que olhar na tabela e encontrar.

algunas colunas que poderia ter utilizado como filtro para obter apenas estes drinks.



Amar

Comparar  
avaliam tu  
uma const

SELE  
FROM  
WHER  
nome  
AND  
nome

Seleci

Foi pedido  
encontrar c

Cada cons  
checa um  
colunas d  
ingredien

## Amarrando os dados em texto com Operadores de Comparação

Comparar dados em texto de um jeito similar com suas colunas de texto como CHAR e VARCHAR. Os operadores de comparação avaliam tudo **alfabeticamente**. Então, digamos, que queira selecionar todos os drinks que começam com a letra 'L', aqui está uma consulta que selecionará todos os drinks com este critério.

`drink_info`

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

```
SELECT nome_do_drink
FROM drink_info
WHERE
  nome_do_drink >= 'L'
  AND
  nome_do_drink < 'M' ;
```

Esta consulta exibe os drinks cuja primeira letra seja L ou posterior, mas que venham antes de M.



Não se preocupe com a ordem dos resultados por enquanto

Em um capítulo posterior nós lhe mostraremos maneiras de separar seus resultados alfabeticamente.

## Selecionando seus ingredientes

Foi pedido a um barman para preparar um drink que tenha suco de cereja nele. O barman poderia utilizar duas consultas para encontrar os coquetéis:

```
File Edit Window Help...
> SELECT nome_do_drink FROM drinks_faceis WHERE principal = 'suco de cereja';
+-----+
| nome_do_drink |
+-----+
| Kiss on the Lips |
+-----+
1 row in set (0.02 sec)

Cada consulta checa um das colunas de ingredientes.

> SELECT nome_do_drink FROM drinks_faceis WHERE segundo = 'suco de cereja';
+-----+
| nome_do_drink |
+-----+
| Lone Tree    |
+-----+
1 row in set (0.01 sec)
```

Ela parece bem  
ineficiente. Tenho certeza  
que deve haver algum jeito de  
unir as duas consultas.



### drink\_info

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

## Ser OR (ou) não ser

Você pode combinar duas consultas utilizando OR (OU). Esta condição exibe registros quando **quaisquer** das condições forem encontradas. Então, ao invés de duas consultas separadas, combine-as com OR (OU) desta forma:

```
File Edit Window Help.. SweetCherryPie
> SELECT nome_do_drink from drinks_faceis
WHERE principal = 'suco de cereja'
OR
segundo = 'suco de cereja';
+-----+
| nome_do_drink |
+-----+
| Kiss on the Lips |
| Lone Tree |
+-----+
2 rows in set (0.02 sec)
```



Aponte seu lápis —

Risque as partes desnecessárias dos dois comandos SELECT abaixo e adicione um OR para torná-las um único comando SELECT.

```
SELECT nome_do_drink FROM drinks_faceis WHERE
principal = 'suco de laranja';
```

```
SELECT nome_do_drink FROM drinks_faceis WHERE
principal = 'suco de maçã';
```

Use suas novas habilidades em selecionar com seu novo SELECT.

.....



Aponte seu lápis —

Solução

Risque as partes desnecessárias dos dois comandos SELECT abaixo e adicione um OR para torná-las um único comando SELECT.

```
SELECT nome_do_drink FROM drinks_faceis WHERE
principal = 'suco de laranja' ,
```

→ OR

```
SELECT nome_do_drink FROM drinks_faceis WHERE
principal = 'suco de maçã';
```

Com o OR nós obtivemos  
os drinks com ingredientes  
principais de suco de laranja e  
suco de maçã.

Use suas novas habilidades em selecionar com seu novo SELECT.

Precisamos nos livrar daquele  
ponto e vírgula para que a  
instrução não termine ainda.

Nós podemos simplesmente  
eliminar esta linha, já cobrimos  
a necessidade dela através da  
primeira parte da consulta  
(agora agregada pelo OR).

```
SELECT nome_do_drink FROM drinks_faceis
```

WHERE

principal = 'suco de laranja'

OR

principal = 'suco de maçã' ;

Agora está a consulta final.



OR parece ser um operador bem útil, mas não entendo porque simplesmente não utilizamos AND.

### Não confunda seus ANDs com ORs!

Quando quiser que **todas** as suas condições sejam verdadeiras use **AND**. Quando quiser que **quaisquer** de suas condições sejam verdadeiras, use **OR**.

# AND

(E)

# OR

(OU)

*não existem*  
Perguntas Idiotas

P: Você pode utilizar mais que um AND ou OR na sua cláusula WHERE?

R: Com certeza! Você pode combinar o quanto quiser. Pode inclusive usar ambos os operadores na mesma cláusula.

## A diferença entre AND e OR

Nas consultas abaixo você verá exemplos de todas as possíveis combinações de duas condições utilizando AND e OR entre elas.

### notas\_donuts

Empresa	horario	data	tipo	nota	comentarios
Krispy King	8:50	27/09	cobertura simples	10	quase perfeito
Duncan's Donuts	8:59	25/08	NULL	6	gorduroso
Starbuzz Coffee	19:35	24/05	bolo de canela	5	não são fresquinhos, mas são gostosos
Duncan's Donuts	19:03	26/04	geléia	7	sem geléia suficiente

```
SELECT tipo FROM notas_donuts
```

*existe coincidência*

```
WHERE empresa = 'Krispy King' AND nota = 10;
```

*sim*

```
WHERE empresa = 'Krispy King' OR nota = 10;
```

```
WHERE empresa = 'Krispy King' AND nota = 3;
```

*Sem coincidência*

```
WHERE empresa = 'Krispy King' OR nota = 3;
```

*Sem coincidência*

```
WHERE empresa = 'Snappy Bagel' AND nota = 10;
```

```
WHERE empresa = 'Snappy Bagel' OR nota = 10;
```

```
WHERE empresa = 'Snappy Bagel' AND nota = 3;
```

```
WHERE empresa = 'Snappy Bagel' OR nota = 3;
```

## RESULTADOS

cobertura simples

cobertura simples

sem resultados

cobertura simples

sem resultados

cobertura simples

sem resultados

sem resultados



## Seja a Condicional

Abaixo você encontrará uma série de cláusulas WHERE com ANDs e ORs. Sinta-se como estas cláusulas e determine se elas vão ou não produzir resultados.

```
SELECT tipo FROM notas_donuts
```

```
WHERE empresa = 'Krispy King' AND nota <> 6;
```

Você conseguiu  
algum resultado?

```
WHERE empresa = 'Krispy King' AND nota = 3;
```

```
WHERE empresa = 'Snappy Bagel' AND nota >= 6;
```

```
WHERE empresa = 'Krispy King' OR nota > 5;
```

```
WHERE empresa = 'Krispy King' OR nota = 3;
```

```
WHERE empresa = 'Snappy Bagel' OR nota = 6;
```

Para aumentar seu potencial, escreva abaixo o porque de dois dos seus resultados são diferentes dos demais.



## Solução de Seja a Condicional

Abaixo você encontrará uma série de cláusulas WHERE com ANDs e ORs. Torne-se um com estas cláusulas e determine se elas vão ou não produzir resultados.

```
SELECT tipo FROM notas_donuts
WHERE empresa = 'Krispy King' AND nota <> 6;
WHERE empresa = 'Krispy King' AND nota = 3;
WHERE empresa = 'Snappy Bagel' AND nota >= 6;
WHERE empresa = 'Krispy King' OR nota > 5;
WHERE empresa = 'Krispy King' OR nota = 3;
WHERE empresa = 'Snappy Bagel' OR nota = 6;
```

Para aumentar seu potencial, escreva abaixo o porque de dois dos seus resultados são diferentes dos demais.

*Duas consultas retornaram o resultado NULL*

Futuramente, aqueles valores NULL poderão causar problemas. É melhor colocar algum tipo de valor do que deixar o valor NULL na coluna **porque o NULL não pode ser diretamente selecionado em uma coluna.**

## Usando IS NULL para encontrar NULL



Tentei selecionar valores NULL diretamente, mas não funcionou. Como eu encontro os valores NULL nas minhas tabelas?

**drink\_info**

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Holiday	NULL	14	NULL	S	50
Dragon Breath	2.9	7.2	marrom	N	NULL

Você não consegue selecionar um valor **NULL** diretamente.

```
SELECT nome_do_drink FROM drink_info
WHERE
calorias = NULL; → Não vai funcionar
porque nada pode ser igual a NULL. É um valor indefinido.

SELECT nome_do_drink FROM drink_info
WHERE
calorias = 0; → Não vai funcionar
porque NULL não é a mesma coisa que zero.

SELECT nome_do_drink FROM drink_info
WHERE
calorias = 'NULL'; → Este não vai
funcionar também,
porque NULL não é uma linha de texto.
```

Mas você pode selecioná-lo usando palavras chave.

```
SELECT nome_do_drink
FROM drink_info
```

```
WHERE
calorias IS NULL;
```

↑ → Palavras-chave  
não são linhas de texto, portanto não precisam de aspas.

A única forma de selecionar diretamente um valor **NULL** é utilizando a expressão **IS NULL**.

## não existem Perguntas Ídiotas

P: Você diz que não consegue “selecionar diretamente” **NULL** sem usar **IS NULL**. Isto quer dizer que consegue selecioná-las indiretamente?

R: Correto. Se quiser chegar ao valor naquela coluna, você poderia utilizar a cláusula WHERE em uma das outras colunas. Por exemplo, seu resultado será **NULL** se utilizar esta consulta:

```
SELECT calorias FROM drink_info
WHERE nome_do_drink = 'Dragon Breath';
```

P: Como os resultados daquela minha consulta seriam?

R: Ela seria exatamente como esta:

```
+-----+
| calories |
+-----+
| NULL     |
+-----+
```

## Enquanto isso, na casa do Greg...

Greg está tentando encontrar todas as pessoas das cidades da Califórnia na sua tabela **meus\_contatos**. Aqui está parte da consulta na qual ele está trabalhando:

Digitar todas essas cláusulas OR é exaustivo!



```
SELECT * FROM meus_contatos
WHERE
local = 'San Fran, CA'
OR
local = 'San Francisco, CA'
OR
local = 'San Jose, CA'
OR
local = 'San Mateo, CA'
OR
local = 'Sunnyvale, CA'
OR
local = 'Marin, CA'
OR
local = 'Oakland, CA'
OR
local = 'Palo Alto, CA'
OR
local = 'Sacramento, CA'
OR
local = 'Los Angeles, CA'
OR
E a lista continua...
```

Ele sabe que digitou San Francisco destas duas maneiras, e quanto aos erros de digitação?

## Economizando tempo com uma simples palavra-chave: LIKE

Há simplesmente cidades demais e ainda mais suas variações de digitação. Usar todas aquelas cláusulas OR irá tirar muito tempo Greg. Felizmente, há uma palavra-chave que economiza muito tempo - LIKE - que, usada como um coringa, procura por parte de uma linha de texto e exibe qualquer resultado compatível.

Greg pode usar LIKE desta forma:

```
SELECT * FROM meus_contatos
WHERE local LIKE '%CA';
```

Colegue um sinal de por cento dentro das aspas simples. Isto diz ao seu Sistema SQL que você está procurando todos os valores na coluna local que terminem com CA.

### O chamado do Coringa

LIKE se agrupa com dois caracteres coringas. Coringas são substitutos para caracteres que existem. Um caractere coringa é igual a qualquer outro caractere em uma linha de texto.



Musculação  
cerebral

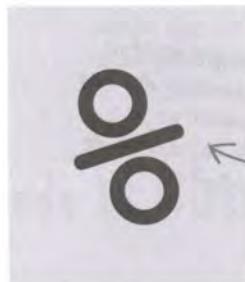
Você viu outros coringas no início deste capítulo?

Coringas são caracteres substitutos.



### É LIKE (como) eu gosto

LIKE gosta de brincar com coringas. O primeiro é o sinal de por cento, %, que pode substituir qualquer número de caracteres desconhecidos.



```
SELECT primeiro_nome FROM meus_contatos
WHERE primeiro_nome LIKE '%im';
```

O por cento é um substituto para qualquer número de caracteres desconhecidos.

Resultará em nomes com quaisquer quantidades de caracteres desconhecidos antes de "im" como Efraim, Slim e Tim.

O segundo caractere coringa que LIKE gosta de ficar junto é a sublinha, \_, que entende apenas um caractere desconhecido.



```
SELECT primeiro_nome FROM meus_contatos
WHERE primeiro_nome LIKE '_im';
```

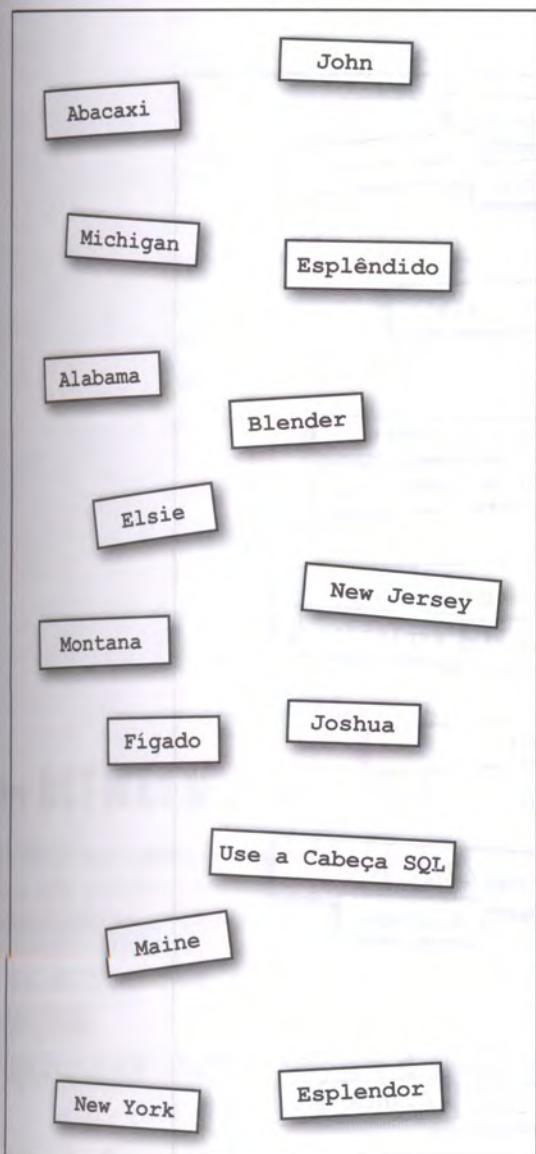
A sublinha é um substituto para apenas um caractere desconhecido.

Resulta em nomes com apenas um caractere antes de "im" como Kim e Tim.



## Ímã de geladeira - compatibilidade

Uma porção de cláusulas WHERE está misturada na porta da geladeira. Você pode unir as cláusulas com seus respectivos resultados? Algumas poderão ter mais do que uma resposta. Escreva seus próprios comandos LIKE com coringas para qualquer resultado que ficar sobrando de lado.



`WHERE estado LIKE 'New %';`

`WHERE nome_da_vaca LIKE '__sie';`

`WHERE titulo LIKE 'USE A CABEÇA%';`

`WHERE rima LIKE '%tender';`

`WHERE primeiro_nome LIKE 'Jo%';`



## Solução de Ímã de geladeira - compatibilidade

Uma porção de cláusulas WHERE está misturada na porta da geladeira. Você pode unir as cláusulas com seus respectivos resultados? Algumas poderão ter mais que uma resposta. Escreva seus próprios comandos LIKE com coringas para qualquer resultado que ficar sobrando de lado.

```
WHERE estado LIKE 'New %';
      New Jersey
      New York
```

```
WHERE nome_da_vaca LIKE '__sie';
      Elsie
```

```
WHERE title LIKE 'HEAD FIRST%';
      Use a Cabeça SQL
```

```
WHERE palavra LIKE 'Sp! %';
      Esplêndid Esplendor
```

```
WHERE rima LIKE '%ender';
      Blender
```

```
WHERE estado LIKE 'M%' OR estado LIKE 'A%';
      Michigan
      Montana
      Maine
      Alabama
```

```
WHERE primeiro_nome LIKE 'Jo%';
      John
      Joshua
```

```
WHERE palavra LIKE '_i%';
      Abacaxi
      Fígado
```

Sele

As pessoas  
consultarão

Bem B

Nós podemos  
também exi  
equivalenteSEL  
WHE  
caleEla inclui o  
30 e 60

## Selecionando alcance usando AND e operadores de comparação

As pessoas do Bar Use a Cabeça estão tentando selecionar drinks com uma certa quantidade de calorias. Como eles consultarão os dados para encontrar nomes de drinks que caberão na faixa de calorias entre 30 e 60?

**drink\_info**

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

`SELECT nome_do_drink FROM drink_info`

`WHERE`

`calorias >= 30`

Os resultados incluirão drinks com calorias igual a 30, se houver algum, aqueles com 60 calorias, bem como os

`AND`

`calorias <= 60`

drinks com calorias dentro desta faixa.

## Bem BETWEEN (entre) nós... há um jeito melhor

Nós podemos usar a palavra-chave BETWEEN ao invés do exemplo anterior. Ela não só é melhor que a consulta anterior, mas também exibe os mesmos resultados. Perceba que o parâmetro inicial e final também estão incluídos (30 e 60). BETWEEN é o equivalente a usar `<=` e `>=`, mas sem o símbolo `< E >`.

`SELECT nome_do_drink FROM drink_info`

`WHERE`

`calorias BETWEEN 30 AND 60;`

Este comando exibirá os mesmos resultados que a consulta na página anterior, mas observe o quanto é mais rápido para digitá-lo!

Ele inclui os drinks com 30 e 60 calorias.

File Edit Window Help MediumCalories

`SELECT nome_do_drink FROM drink_info`

`WHERE`

`calorias BETWEEN 30 AND 60;`

+	-----+
	nome_do_drink
+	-----+
	Blackthorn
	Oh My Gosh
	Greyhound
	Indian Summer
	Soda and It
+	-----+



## Aponte seu lápis —

Reescreva a consulta da página anterior para SELECT (selecionar) todos os nomes de drinks que tenham mais que 60 calorias e menos que 30.

.....  
.....  
.....

Tente usar BETWEEN nas colunas de texto. Escreva uma consulta que irá SELECT (selecionar) os nomes dos drinks que começam com a letra G até O.

.....  
.....  
.....

Como você acha que serão os resultados desta consulta?

```
SELECT nome_do_drink FROM drink_info WHERE  
calorias BETWEEN 60 AND 30;
```



## Aponte seu lápis —

## Solução

Reescreva a consulta da página anterior para SELECT (selecionar) todos os nomes de drinks que tenham mais que 60 calorias e menos que 30.

*SELECT nome do drink FROM drink\_info  
WHERE  
calorias < 30 OR calorias > 60;*

*Nós dímos nomes de drinks com calorias maiores que 60.*

*Estes aqui são com calorias menores que 30.*

Tente usar BETWEEN nas colunas de texto. Escreva uma consulta que irá SELECT (selecionar) os nomes dos drinks que começam com a letra G até O.

*SELECT nome do drink FROM drink\_info  
WHERE  
nome do drink BETWEEN 'G' AND 'O';*

*Conseguiremos nomes de drinks que começam com G e O, e todas as letras entre elas.*

Como você acha que serão os resultados desta consulta?

```
SELECT nome_do_drink FROM drink_info WHERE  
calorias BETWEEN 60 AND 30;
```

*A ordem importa! Então você não obterá resultado algum com esta consulta.*

Nós estamos procurando valores que estão entre 60 e 30. Não há valores entre 60 e 30, porque 60 vem antes que 30 numericamente. O número menor deve sempre ficar primeiro para que o **BETWEEN** seja interpretado do jeito que espera.

Depo

A amiga d  
mantém i

Ela nome

SEL  
FRO  
WHE  
ava  
OR  
ava  
OR  
... ;

Ao invés d  
IN com ur  
a um dos v

SEL  
FRO  
WHE  
ava  
'fab  
'mui

...OU

É claro, An  
que está lav  
Para encon  
a palavra-cl  
resultado n

## Depois dos encontros, você está IN...

A amiga de Greg, Amanda tem usado os contatos dele para encontrar rapazes. Ela já foi em alguns encontros, e até já criou uma tabela “livro negro” com suas impressões sobre seus encontros.

Ela nomeou sua tabela de `livro_negro`. Ela quer obter uma lista dos bons encontros, então ela dá avaliações positivas.

```
SELECT nome_par
FROM livro_negro
WHERE
    avaliacao = 'inovador'
OR
    avaliacao = 'fabuloso'
OR
    ...;
```

Você precisa de uma linha para cada ponto positivo.

Estas são as avaliações positivas.

### livro\_negro

nome_par	avaliacao
Alex	inovador
James	chato
Ian	fabuloso
Boris	oh-oh
Melvin	plebeu
Eric	patético
Anthony	prazeroso
Sammy	muito bom
Ivan	deplorável
Vic	ridículo

Ao invés de utilizar todos aqueles ORs, podemos simplificar com a palavra-chave IN. Use IN com um conjunto de valores em parênteses. Quando o valor na coluna for compatível com um dos valores do conjunto, a linha ou coluna específica é exibida.

```
SELECT nome_par
FROM livro_negro
WHERE
    avaliacao IN ('inovador',
    'fabuloso', 'prazeroso',
    'muito bom');
```

Ao usar a palavra-chave IN, ela diz ao Sistema SQL que um conjunto de valores está vindo.

Este é o conjunto de avaliações positivas.

```
File Edit Window Help GoodDates
> SELECT nome_par FROM livro_negro
WHERE
    avaliacao IN ('inovador', 'fabuloso',
    'prazeroso', 'muito bom');

+-----+-----+
| nome_par |
+-----+
| Alex     |
| Ian      |
| Anthony |
| Sammy   |
+-----+
```

## ...ou você está NOT IN

Claro, Amanda quer saber quem tem más avaliações para quando eles ligarem ela dizer que está lavando o cabelo ou compromissada com alguém.

Para encontrar o nome daqueles que ela não avaliou de forma boa, iremos adicionar a palavra-chave NOT na nossa declaração IN. NOT dá os resultados opostos, qualquer resultado não compatível ao conjunto.

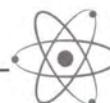
```
SELECT nome_par
FROM livro_negro
WHERE
    avaliacao NOT IN ('inovador',
    'fabuloso', 'prazeroso',
    'muito bom');
```

Usando a palavra-chave NOT IN, ela diz ao Sistema SQL que os resultados não estão no conjunto de termos.



```
File Edit Window Help BadDates
> SELECT nome_par FROM livro_negro
  WHERE
    avaliacao NOT IN ('inovador', 'fabuloso',
    'prazeroso', 'muito bom');
+-----+
| nome_par |
+-----+
| James   |
| Boris   |
| Melvin  |
| Eric    |
| Ivan    |
| Vic    |
+-----+
6 rows in set (2.43 sec)
```

Usando o NOT IN para obter pessoas com avaliações negativas.



### PODER DO CÉREBRO

Por que às vezes você vai usar NOT IN ao invés de IN?

## Mais NOT

Você pode usar o NOT com BETWEEN e LIKE da mesma forma que o faz com IN. O mais importante a se lembrar é que **NOT vai logo após WHERE** no seu comando. Aqui estão alguns exemplos:

```
SELECT nome_do_drink FROM drink_info
WHERE NOT carboidratos BETWEEN 3 AND 5;
```

Quando você usa o NOT com AND ou OR, ele vai logo após o AND ou OR.

```
SELECT nome_par from lista_negra
WHERE NOT nome_par LIKE 'A%'
AND NOT nome_par LIKE 'B%';
```

---

não existem  
Perguntas Idiotas

---

P: Espere! Você disse que NOT vai logo após WHERE, e quando você utiliza NOT IN?

R: Esta é uma exceção e mesmo movendo o NOT para depois do WHERE ele vai funcionar. Estes dois comandos lhe darão exatamente os mesmos resultados:

```
SELECT * FROM drinks_faceis
WHERE NOT principal IN ('soda', 'chá
gelado');
```

```
SELECT * FROM drinks_faceis
WHERE principal NOT IN ('soda', 'chá
gelado');
```

P: Ela funcionaria com  $\neq$  o "não igual" operador de comparação?

R: Você poderia, mas é uma negativa dupla. Faria muito mais sentido apenas utilizar o sinal de igual. Estas duas consultas exibem o mesmo resultado:

```
SELECT * FROM drinks_faceis
WHERE NOT nome_do_drink <> 'Blackthorn';
```

```
SELECT * FROM drinks_faceis
WHERE nome_do_drink = 'Blackthorn';
```

P: Como ela funcionaria com NULL?

R: Do jeito que imagina que ele iria. Vai buscar todos os valores que não são NULL em uma coluna, você poderia utilizá-lo assim:

```
SELECT * FROM drinks_faceis
WHERE NOT principal IS NULL;
```

Mas isto também funcionaria:

```
SELECT * FROM drinks_faceis
WHERE principal IS NOT NULL;
```

P: E quanto ao AND ou OR?

R: Se quisesse utilizá-lo em uma cláusula AND ou OR, ela iria logo após a palavra, desta forma:

```
SELECT * FROM drinks_faceis
WHERE NOT principal = 'soda'
AND NOT principal = 'chá gelado';
```



## Exercícios

Reescreva cada uma das cláusulas WHERE para que fiquem o mais simples possível. Você pode utilizar AND, OR, NOT, BETWEEN, LIKE, IN, IS NULL, e os operadores de comparação para lhe ajudar.

Use como referências as tabelas estudadas neste capítulo.

```
SELECT nome_do_drink from drinks_faceis  
WHERE NOT quantidade1 < 1.50;
```

```
SELECT nome_do_drink FROM drink_info  
WHERE NOT gelo = 'S';
```

```
SELECT nome_do_drink FROM drink_info  
WHERE NOT calorias < 20;
```

```
SELECT nome_do_drink FROM drinks_faceis  
WHERE principal = 'néctar de pêssego'  
OR principal = 'soda';
```

```
SELECT nome_do_drink FROM drink_info  
WHERE NOT calorias = 0;
```

```
SELECT nome_do_drink FROM drink_info  
WHERE NOT carboidratos BETWEEN 3 AND 5;
```

```
SELECT nome_par from livro_negro  
WHERE NOT nome_par LIKE 'A%'  
AND NOT nome_par LIKE 'B%';
```



## Solução dos Exercícios

Reescreva cada uma das cláusulas WHERE para que fiquem o mais simples possível. Você pode utilizar AND, OR, NOT, BETWEEN, LIKE, IN, IS NULL, e os operadores de comparação para lhe ajudar.

Use como referências as tabelas estudadas neste capítulo.

```
SELECT nome_do_drink from drinks_faceis
WHERE NOT quantidade1 < 1.50;
```

```
.....SELECT nome do drink from drinks faceis
```

```
.....WHERE quantidade1 >= 1.50;
```

```
SELECT nome_do_drink FROM drink_info
WHERE NOT gelo = 'S' ;
```

```
.....SELECT nome do drink FROM drink info
```

```
.....WHERE gelo = 'N';
```

```
SELECT nome_do_drink FROM drink_info
WHERE NOT calorias < 20;
```

```
.....SELECT nome do drink FROM drink info
```

```
.....WHERE calorias >= 20;
```

```
SELECT nome_do_drink FROM drinks_faceis
WHERE principal = 'néctar de pêssego'
```

```
OR principal = 'soda' ;
```

```
.....SELECT nome do drink FROM drinks faceis
```

```
.....WHERE principal BETWEEN 'P' AND 'S';
```

Este comando só funcionará porque não temos outro ingrediente principal que satisfaça a essa condição. Se nossa tabela tivesse suco de nectarina, ele não funcionaria.

```
SELECT nome_do_drink FROM drink_info
WHERE NOT calorias = 0;
```

```
.....SELECT nome do drink FROM drink info
```

```
.....WHERE NOT calorias > 0;
```

Nós nunca teremos calorias negativas, então temos uma consulta segura com o sinal de maior:

```
SELECT nome_do_drink FROM drink_info
WHERE NOT carboidratos BETWEEN 3 AND 5;
```

```
.....SELECT nome do drink FROM drink info
```

```
.....WHERE carboidratos < 3
```

OR

Carboidratos > 5;

```
SELECT nome_par from livro_negro
```

```
WHERE NOT nome_par LIKE 'A%'
```

```
AND NOT nome_par LIKE 'B%' ;
```

```
.....SELECT nome_par from livro_negro
```

```
.....WHERE nome_par NOT BETWEEN 'A' AND 'B';
```



## Sua caixa de ferramenta SQL

Você já tem o capítulo 2 na palma da mão e adicionou operadores na sua caixa de ferramentas. Para uma lista completa de dicas de ferramentas, veja o Apêndice iii.

`SELECT *`

use este comando para selecionar tudo na tabela.

Escape com ` e \

Escape dos apóstrofes nos seus dados de texto usando um apóstrofo extra ou uma barra invertida na frente do sinal.

`= <> < > <= >=`

Você tem uma porção de operadores de igualdade e desigualdade à sua disposição.

`IS NULL`

Use-o para criar uma condição para testar aquele valor `NULL` inoportuno.

`AND` e `OR`

Com `AND` e `OR` você pode combinar códigos condicionais nas suas cláusulas `WHERE` com mais precisão.

`NOT`

`NOT` permite que você negue seus resultados e obtenha apenas os valores apostos.

`BETWEEN`

Permite que você selecione uma certa abrangência de valores.

`LIKE` com % e \_

Use `LIKE` com seus coringas para procurar por partes de linhas de texto.

Suas novas ferramentas!  
operadores!



## Solução dos Exercícios da página 52.

Greg quer criar uma tabela de drinks que os bartenders possam consultar os ingredientes em eventos rápidos. Usando o que aprendeu no capítulo 1, escreva os comandos em SQL e crie a tabela desta página e em seguida, insira os dados mostrados.

A tabela é parte de um banco de dados chamado drinks. Ela contém a tabela drinks\_faceis com receitas para um número de bebidas que tem apenas dois ingredientes.

```
CREATE DATABASE drinks;
USE drinks;
CREATE TABLE drinks_faceis
  (nome_do_drink VARCHAR(16), principal VARCHAR(20), quantidade1
  DEC(3,1), segundo VARCHAR(20), quantidade2 DEC(4,2), instrucoes
  VARCHAR(250));
```

*É uma boa ideia lhe dar alguns caracteres extras no caso de ter que colocar algum nome que seja maior do que estes já existentes.*

```
INSERT INTO drinks_faceis
VALUES
```

*Não se esqueça: tipos de dados numéricos não precisam de aspas!*

```
('Blackthorn', 'Água tônica', 1.5, 'Suco de abacaxi', 1, 'Mexa com gelo,
coloque em uma taça de coquetel com limão batido'), ('Blue moon', 'soda', 1.5,
'suco de mirtilo', .75, 'Mexa com gelo, coloque em uma taça de coquetel com
limão batido'), ('Oh My Gosh', 'néctar de pêssego', 1, 'suco de abacaxi', 1,
'Mexa com gelo, coloque em um copinho de licor'),
('Lime Fizz', 'Sprite', 1.5, 'suco de limão', .75, 'Mexa com gelo, coloque em
uma taça de coquetel'),
('Kiss on the Lips', 'suco de cereja', 2, 'néctar de damasco', 7, 'Sirva com
gelo e canudo'),
('Hot Gold', 'néctar de pêssego', 3, 'suco de laranja', 6, 'coloque suco de
laranja quente numa caneca e adicione néctar de pêssego'),
('Lone Tree', 'soda', 1.5, 'suco de cereja', .75, 'Mexa com gelo, coloque em
uma taça de coquetel'),
('Greyhound', 'soda', 1.5, 'suco de toranja', 5, 'Sirva com gelo, mexa bem'),
('Indian Summer', 'suco de maçã', 2, 'chá quente', 6, 'adiciona o suco a uma
caneca e complete o resto com chá quente'),
('Bull Frog', 'chá gelado', 1.5, 'limonada', 5, 'Sirva com gelo e uma fatia de lima'),
('Soda and it', 'soda', 2, 'suco de uva', 1, 'misture em uma taça de coquetel,
sem gelo');
```

*Todo o conjunto de valores referente a um drink está entre parênteses.*

*E entre cada drink há uma vírgula.*

### DICA:

O script que cria esta tabela pode ser baixada em nosso site: [www.altabooks.com.br](http://www.altabooks.com.br)

schieden werden können.  
Zurzeit überwiegen jedoch  
die technologischen  
Faktoren, die maßgeblich an  
der Entwicklung der Branche  
zu sehen sind.

Die technologische Entwicklung ist  
die resultierende Wirkung von  
Technologien, die in der Produktion  
verwendet werden. Die  
Technologien sind hierbei nicht  
nur auf die Produktion beschränkt,  
sondern umfassen auch die  
Logistik und die Marketing-  
und Vertriebsaktivitäten.

Die technologische Entwicklung  
ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.  
Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

Die technologische Entwicklung ist eine wichtige Faktoren, die die  
Entwicklung der Branche beeinflusst.

## 3 DELETE e UPDATE

# Uma mudança na sua vida fará bem a você



**Você muda de opinião constantemente?** Agora está tudo bem! Com os comandos que você está prestes a conhecer – DELETE e UPDATE – você não mais precisará ficar preso a uma decisão que tomou há seis meses atrás, quando inseriu dados dizendo que calça boca-de-sino logo estaria de volta à moda. Com UPDATE, você pode mudar os dados, e DELETE permite eliminar os dados que não precisar mais. Mas não estamos apenas lhe dando as ferramentas; neste capítulo, você aprenderá a ser mais seletivo com seus novos poderes e a evitar que suma com arquivos que realmente precise.

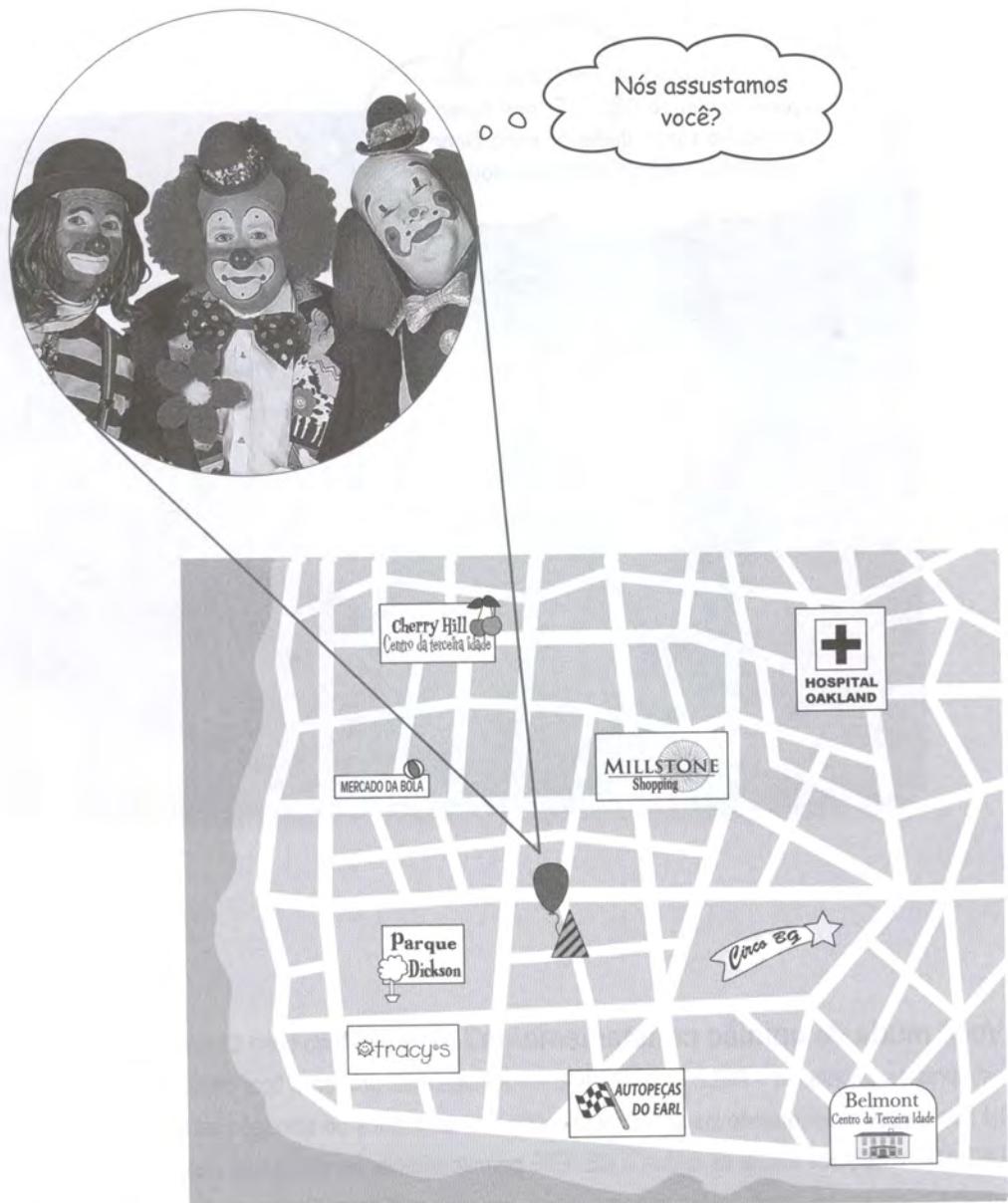
## Palhaços são assustadores?

Suponha que queremos saber onde andam todos os palhaços de festas na cidade de Dataville. Nós poderíamos criar uma tabela `palhaco_info` para rastreá-los. E poderíamos criar a coluna `visto_ultimo` para guardar onde eles foram vistos pela última vez.

Rastre

Aqui está n  
com.br). N  
que tiverme  
que alterar

Quand  
avista



## Rastreador de palhaços

Aqui está nossa tabela (que você pode criar facilmente baixando o script do site [www.altabooks.com.br](http://www.altabooks.com.br)). Nós podemos deixar as informações que não sabemos e preenchê-las depois. Toda vez que tivermos uma nova visão de algum palhaço poderemos adicionar uma nova linha. Teremos que alterar esta tabela freqüentemente para mantê-la atualizada.

*Quando cada palhaço foi visto pela última vez.*

**palhaco\_info**

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado de Bolas	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	



*Preencheremos os espaços em brancos depois.*



Aponte seu lápis

## Os palhaços estão em movimento

Seu trabalho é escrever os comandos SQL para conseguir o relatório de cada área na tabela palhaco\_info. Perceba que nem todas as informações mudaram para cada palhaço, então você terá que usar a tabela na página 99 como referência para conseguir as demais informações a serem adicionadas.

Zippo foi visto cantando

INSERT INTO palhaco\_info

VALUES

('Zippo', 'Shopping', 'Millstone', 'F', 'ternos laranja, calças grandes',  
'dançar, cantar');

Snuggles agora está vestindo  
calças largas azuis.

INSERT INTO palhaco\_info

VALUES

('Snuggles', 'Mercado da Bala', 'F', 'camiseta amarela, calças largas  
azuis', 'cornete, guarda-chuva');

Bonzo avistado no  
Parque Dickson

Snuffles visto montando em  
um carrinho

Mr. Hobo visto por último na  
Festa de Eric Gray

Agora preencha da forma como a tabela palhaco\_info se parece  
após você ter adicionado mais dados usando o comando INSERT.

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado de Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Snaffles	Tracy's	M, terno verde e roxo, nariz pontudo	



Aponte seu lápis  
Solução

## Os palhaços estão em movimento

Seu trabalho é escrever os comandos SQL para conseguir o relatório de cada área na tabela palhaco\_info. Perceba que nem todas as informações mudaram para cada palhaço, então você terá que usar a tabela na página 99 como referência para conseguir as demais informações a serem adicionadas.

Zippo foi visto cantando

`INSERT INTO palhaco_info`

`VALUES`

```
('Zippo', 'Shopping Millstone', 'F', 'jornal laranja, calças grandes',
'dançar, cantar');
```

Snuggles agora está vestindo  
calças largas azuis.

`INSERT INTO palhaco_info`

`VALUES`

```
('Snuggles', 'Mercado da Bola', 'F', 'camiseta amarela,
calças largas azuis', 'cornete, guarda-chuva');
```

Bonzo avistado no  
Parque Dickson

`INSERT INTO palhaco_info`

`VALUES`

```
('Bonzo', 'Parque da Dickson', 'M', 'vestido comprido de bolinhas',
'dançar, cantar');
```

Snuffles visto montando em  
um carrinho

`INSERT INTO palhaco_info`

`VALUES`

```
('Snuffles', 'Tracy\'s', 'M', 'jornal verde e roxo, nariz pontudo',
'montando em um carrinho');
```

Mr. Hobo visto por último na  
Festa de Eric Gray

`INSERT INTO palhaco_info`

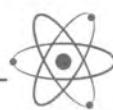
`VALUES`

```
('Mr. Hobo', 'Festa de Eric Gray', 'M', 'charuto, cabelo preto,
chapéu comprido', 'violino');
```

Não esqueça de escapar as aspas nos  
seus valores VARCHAR

Agora preencha da forma como a tabela palhaco\_info se parece após você ter adicionado mais dados usando o comando INSERT.

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado de Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo	Parque do Dickson	M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	montando em um carrinho
Zippo	Shopping Millstone	F, terno laranja, calças grandes	dançar, cantar
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas aquis	corneta, guarda-chuva
Bonzo	Parque do Dickson	M, vestido comprido de bolinhas	dançar, cantar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	montando em um carrinho
Mr. Hobo	Festa de Eric Gray	M, charuto, cabelo preto, chapéu comprido	violino



**PODER DO  
CÉREBRO**

Como você pode descobrir a localização atual de um palhaço?

# Como os nossos dados de palhaços são inseridos?

Nossos localizadores de palhaços trabalham de forma voluntária. Às vezes, eles ficam à espera de novas informações por uma ou duas semanas antes de receber novos dados. Às vezes duas pessoas **dividem a pilha** de registros e os **inserem ao mesmo tempo**. Tendo isso em mente, vamos olhar todas as linhas na nossa tabela referente ao palhaço Zippo. Podemos usar o comando SELECT para obter as respostas.

```
File Edit Window Help CatchTheClown
SELECT * FROM palhaco_info WHERE nome = 'Zippo';
```

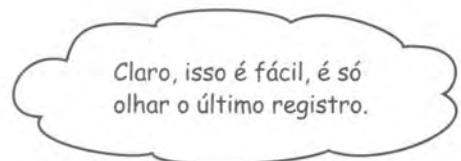
nome	visto_ultimo	aparencia	atividades
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar
Zippo	Hospital Oakland	F, terno laranja, calças largas	dançar, cantar
Zippo	Tracy's	F, terno laranja, calças largas	dançar, cantar
Zippo	Mercado da Bola	F, terno laranja, calças largas	dançar, malabarismo
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar
Zippo	Hospital Oakland	F, terno laranja, calças largas	dançar, cantar

Estes dois registros são exatamente idênticos.

Estes dois também são idênticos.

Estas informações ficam se repetindo.

Há alguma maneira de fazer uma consulta em nossos dados e obter apenas os registros mais recentes da visualização de Zippo? Você consegue dizer qual foi a sua última localização.



## Infelizmente não é possível dizer se o último registro é o mais atual.

Nós temos mais de uma pessoa inserindo dados ao mesmo tempo. E os registros podem ter se misturado. Mas ainda que esse fosse o problema, **você não pode confiar que as linhas da tabela estejam em ordem cronológica**.

Há diversos fatores internos no banco de dados que poderiam alterar a ordem em que as linhas são armazenadas. Isto inclui se o seu Sistema SQL (RDBMS) indexa as colunas (sobre o que falaremos posteriormente).

Você não pode garantir que a última linha da tabela seja a linha mais recente adicionada à tabela.

Bozo

Já que n  
nos dá i  
**cada p**

E isso n  
fazendo  
maior. F  
porque c  
que nun  
tabela já

P: R:

os m  
você  
se p  
últim  
tirar  
INSE  
para  
dato  
inser

P: s

R: v  
por a  
inser  
de re  
de re  
esper

## Bozo, nós temos um problema

ma ou  
no tempo.  
SELECT

que não se pode contar como a última linha sendo o registro mais recente, temos um problema. Nossa tabela de palhaços

da lista de onde os palhaços estariam em certo momento. **Mas a razão principal da tabela é saber onde cada palhaço foi visto pela última vez.**

Isso não é tudo. Você notou aqueles registros duplicados? Nós temos dois registros mostrando Zippo no mesmo lugar, tendo a mesma coisa. Eles ocupam espaço e vão desacelerar seu Sistema SQL na medida em que sua tabela fica cada vez maior. Registros duplicados **não deveriam existir nunca em uma tabela**. Em poucos capítulos estaremos falando porque os registros duplicados são ruins e como evitá-los com um **bom projeto de tabela**. Você verá como criar tabelas que nunca terão registros duplicados. Mas, no momento, vamos nos concentrar no que podemos fazer para consertar nossa tabela já existente para que assim ela contenha dados úteis.

### *não existem* Perguntas Ídiotas

P: Por que não podemos presumir que os últimos registros são os mais recentes?

R: A ordem dos registros em uma tabela não é garantida e em breve você vai modificando a ordem dos resultados conforme obtém. Não se pode ter confiança absoluta que o último registro é realmente o último registro inserido, bem como simples erros humanos poderiam tirar a ordem da tabela. Suponhamos que inserimos dois comandos INSERT para o mesmo palhaço, exceto façamos alguma observação para indicar que um registro venha antes do outro, depois que estes dados estiverem em sua tabela, não saberemos com certeza qual foi inserido primeiro.

P: Suponhamos que sabemos sim a ordem. Novamente, por que não poderíamos simplesmente usar o último registro?

R: Vamos ampliar o exemplo. Temos rastreado o mesmo palhaço por anos. Talvez tenhamos assistentes que rastreiem e que também inserem suas próprias informações. Alguns dos palhaços têm centenas de registros. Quando usamos o SELECT, obtemos aquelas centenas de registros pelas quais temos que nos arrastar até o último, que esperançosamente esperamos ser a mais recente.

P: Não há momentos quando queremos manter dados assim na tabela? Faz sentido INSERT (inserir) novos registros e manter os antigos?

R: Absolutamente. Use o nosso exemplo atual. A tabela como está agora, não só nos informa o último lugar em que um palhaço foi avistado, mas também dá um histórico de seus movimentos. Essa é uma informação útil em potencial. O problema é que não temos nenhuma informação clara em cada registro que nos diga quando o fato ocorreu. Se adicionarmos uma coluna com a data e a hora atual, talvez, estariamos aptos a rastrear os palhaços com grande exatidão.

Mas no momento precisamos conseguir remover aqueles próximos registros duplicados da nossa tabela para simplificar as coisas.

P: Ok, então até o final do livro eu saberei como projetar tabelas sem linhas duplicadas. O que acontece se o cara que trabalhou antes de mim, deixou uma tabela mal projetada?

R: Tabelas mal projetadas são comuns no mundo real, mas a maioria das pessoas que aprendem SQL descobre que tem que consertar a bagunça na SQL feita por outras pessoas.

Há uma porção de técnicas para limpar as linhas duplicadas. Algumas das melhores envolvem joins, um tópico a ser visto posteriormente neste livro. Até o momento, você não tem todas as ferramentas que precisa para consertar dados ruins, mas as terá no final.

po?

a  
l.

## Eliminando um registro com DELETE

Parece que teremos que eliminar (também conhecido como “deletar”) alguns registros. Para que a tabela nos seja mais útil, deveríamos ter somente uma linha por palhaço. Enquanto esperamos por uma nova visualização de Zippo chegar, uma que de fato saberemos que é a mais recente, nós poderemos eliminar alguns dos registros antigos de Zippo que não nos ajuda.

O comando **DELETE** é sua ferramenta para deletar linhas de dados de sua tabela. Ela utiliza o mesmo tipo de cláusula que WHERE que você já viu.

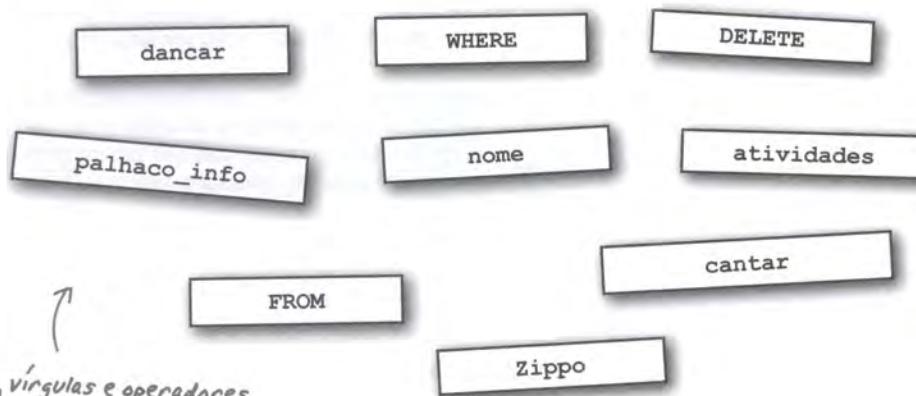
Aqui estão as linhas para Zippo novamente:

nome	visto_ultimo	aparencia	atividades
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar
Zippo	Hospital Oakland	F, terno laranja, calças largas	dançar, cantar
Zippo	Tracy's	F, terno laranja, calças largas	dançar, cantar
Zippo	Mercado da Bola	F, terno laranja, calças largas	dançar, malabarismo
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar
Zippo	Hospital Oakland	F, terno laranja, calças largas	dançar, cantar



### Ímãs de geladeira - comando DELETE

Nós escrevemos um comando simples que poderíamos utilizar para eliminar um dos registros de Zippo, mas todos os ímãs caíram da geladeira. Junte novamente os fragmentos e anote o que acha que cada parte do comando faz.

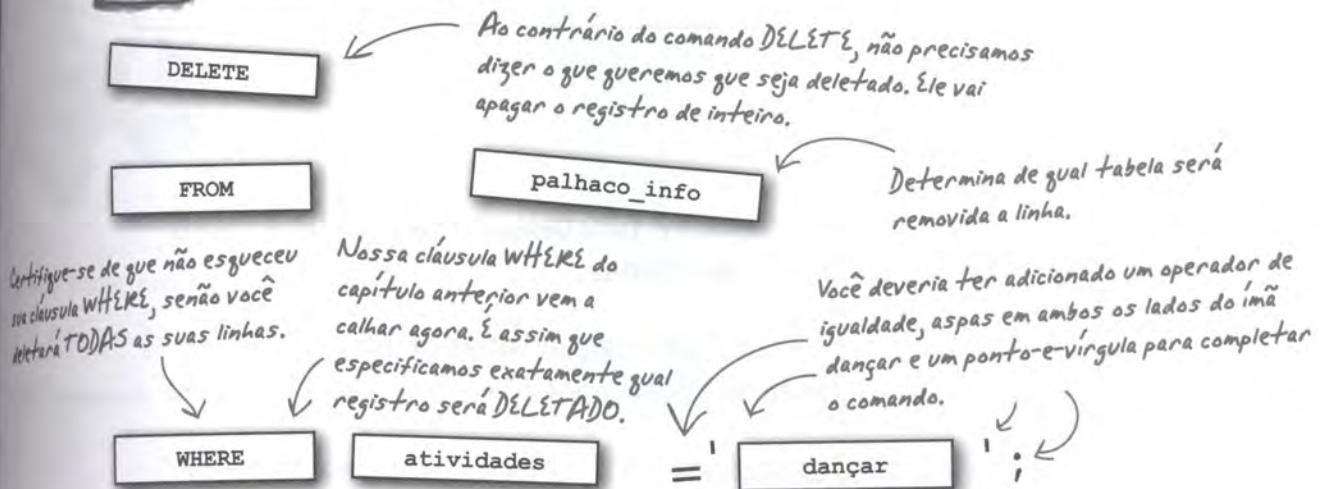


As aspas, vírgulas e operadores de igualdade e ponto-e-vírgulas eram pequenos demais para serem recolhidos do chão. Fique à vontade para adicionar tantos quantos forem necessários.



## Solução de ímãs de geladeira - comando DELETE

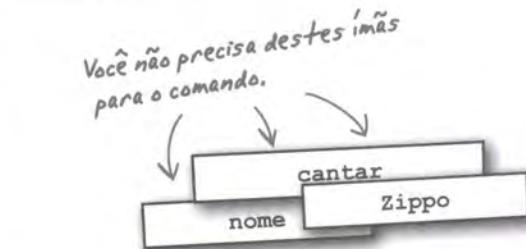
Nós escrevemos um comando simples que poderíamos utilizar para eliminar um dos registros de Zippo, mas todos os ímãs caíram da geladeira. Junte novamente os fragmentos e anote o que acha que cada parte do comando faz.



Você pode usar as cláusulas **WHERE** com declarações **DELETE** da mesma forma que as usou no comando **INSERT**.

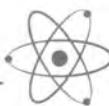
## Usando nossa nova instrução **DELETE**

Vamos usar o comando que acabamos de criar. Isto faz realmente o que parece que ele faria. Todos os registros que se ajustam a uma condição da cláusula **WHERE** serão deletados de sua tabela.



**DELETE FROM palhaco\_info WHERE atividades = 'dançar';**

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar e cantar
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e azuis	corneta, guarda-chuva
Bonzo	Parque do Dickson	M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	montando em um carrinho
Mr. Hobo	Festa de Eric Gray	M, charuto, cabelo preto, chapéu comprido	violino



**PODER DO  
CÉREBRO**

Você acredita que pode deletar uma única coluna de uma linha usando DELETE?

## Regras do DELETE

- Você não pode usar `DELETE` para deletar o valor de uma coluna simples ou uma porção de colunas.
- Você pode usar `DELETE` para deletar linhas simples ou linhas múltiplas dependendo da cláusula `WHERE`.
- Você já viu como apagar uma linha da tabela. Podemos também apagar múltiplas linhas de uma tabela. Para isto, utilizamos uma cláusula `WHERE` para dizer ao `DELETE` quais as linhas escolhidas. Esta cláusula `WHERE` é exatamente a mesma utilizada no capítulo 2 com seus comandos `SELECT`. Ele pode usar todas as ferramentas que você utilizou no capítulo 2, tais como `LIKE`, `IN`, `BETWEEN` e todas as condicionais para dizer precisamente ao seu Sistema SQL, com exatidão, quais linhas serão apagadas.
- E preste bastante atenção nisso: você pode deletar todas as linhas de uma tabela da seguinte forma: `DELETE FROM sua_tabela`.

### *não existem* **Perguntas Idiotas**

**P:** Existe diferença em utilizar o `WHERE` com um comando `DELETE` versus `WHERE` com `SELECT`?

**R:** Nenhuma diferença. O `WHERE` sempre será o mesmo, mas a diferença entre o que o `DELETE` e o `SELECT` fazem é significativa. `SELECT` retorna uma cópia das colunas das linhas que se ajustam a uma condição, mas não altera a sua tabela. `DELETE` remove qualquer linha que se ajusta à condição `WHERE`. Ele remove a linha inteira da tabela.

## Seja o DELETE nas cláusulas WHERE



Sinta-se como uma declaração  
DELETE com cláusulas WHERE,  
seja também AND e ORs para  
determinar ou não algumas das  
linhas.

```
DELETE FROM notas_donuts
```

```
WHERE empresa = 'Krispy King' AND nota <> 6;
```

```
WHERE empresa = 'Krispy King' AND nota = 3;
```

```
WHERE empresa = 'Snappy Bagel' AND nota >= 6;
```

```
WHERE empresa = 'Krispy King' OR nota > 5;
```

```
WHERE empresa = 'Krispy King' OR nota = 3;
```

```
WHERE empresa = 'Snappy Bagel' OR nota = 6;
```

**Risque a linha, ou  
linhas, que cada  
consulta deletou:**

notas\_donuts

Empresa	Horario	data	tipo	nota	comentarios
Krispy King	8:50	27/09	cobertura simples	10	quase perfeito
Duncan's Donuts	8:59	25/08	NULL	6	gorduroso
Starbuzz Coffee	19:35	24/05	bolo de canela	5	Não são fresquinhos, mas são gostosos
Duncan's Donuts	19:03	26/04	geléia	7	Sem geléia suficiente



## Seja o DELETE nas cláusulas WHERE

Sinta-se como uma declaração  
DELETE com cláusulas WHERE,  
seja também AND e ORs para  
determinar ou não algumas das  
linhas.

```
DELETE FROM notas_donuts
```

**Risque a linha, ou  
linhas, que cada  
consulta deletou:**

```
WHERE empresa = 'Krispy King' AND nota <> 6;
```

```
WHERE empresa = 'Krispy King' AND nota = 3; Sem resultado, não  
deletou.
```

```
WHERE empresa = 'Snappy Bagel' AND nota >= 6; Sem resultado, não  
deletou.
```

```
WHERE empresa = 'Krispy King' OR nota > 5;
```

```
WHERE empresa = 'Krispy King' OR nota = 3;
```

```
WHERE empresa = 'Snappy Bagel' OR nota = 6; Sem resultado, não  
deletou.
```

**notas\_donuts**

Empresa	Horario	data	tipo	nota	comentarios
Krispy King	8:50	27/09	cobertura simples	10	quase perfeito
Duncan's Donuts	8:59	25/08	NULL	6	gorduroso
Starbuzz Coffee	19:35	24/05	bolo de canela	5	Não são fresquinhos, mas são gostosos
Duncan's Donuts	19:03	26/04	geléia	7	Sem geléia suficiente

Agueles valores NULL podem lhe causar problemas em consultas futuras. É melhor inserir algum tipo de valor do que ter um valor NULL na coluna, porque os valores NULL não podem ser encontrados com uma condição de igualdade.

## Os dois

Há apenas um  
Já que querer  
informe sua i  
um novo regi

1

INSE

VALU

('Cl  
rosa

INSERT

2

## Os dois passos do INSERT e DELETE

apenas um registro sobre Clarabelle na tabela inteira.

que queremos apenas um palhaço por linha que nos  
ofereça sua informação mais recente, nós precisamos criar  
um novo registro e apagar o velho.

Apenas a atividade dela está  
diferente da que na linha atual.

Clarabelle avistada ~~dançando~~ no Centro de  
Terceira Idade Belmont, F, cabelo rosa, flores  
gigantes, vestido azul.

Nossa trabalho era adicionar os dados nesta  
tabela. Nós estamos mostrando apenas uma  
linha da tabela na página 107 para não ocupar  
tanto espaço.

nome	visto_ultimo	aparencia	atividades
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar

- 1 Primeiro, use o INSERT para adicionar a nova informação (e as informações antigas também).

INSERT INTO palhacos\_info  
VALUES

('Clarabelle', 'Centro da Terceira Idade Belmont', 'F, cabelo  
rosa, flores gigantes, vestido azul', 'dançar');

INSERT (insira) o registro usando todos  
os dados originais apenas alterando a  
coluna que precisar alterar.

nome	visto_ultimo	aparencia	atividades
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar

INSERT	Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	dançar
--------	------------	-------------------------------------	--	--------

- 2 Então, DELETE o registro antigo usando uma cláusula WHERE.

DELETE FROM palhaco\_info  
WHERE  
atividades = 'gritar'  
AND nome = 'Clarabelle';

Use cláusula WHERE para encontrar  
e deletar o registro antigo.

Agora nós temos na tabela apenas o registro novo.

nome	visto_ultimo	aparencia	atividades
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	dançar



Aponte seu lápis

Use INSERT e DELETE para mudar a tabela drink\_info como pedido. Então, desenhe a tabela alterada ao lado.

### drink\_info

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

Mude as calorias de Kiss on the Lips para 170.

.....

.....

.....

.....

Mude os valores do amarelo para dourado.

.....

.....

.....

.....

drink\_info

name_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn					
Blue Moon					
Oh My Gosh					
Lime Fizz					
Kiss on the Lips					
Hot Gold					
Lone Tree					
Greyhound					
Indian Summer					
Bull Frog					
Soda and It					



0 0 Esta é mais uma de suas pegadinhas?

Faça com que todos os drinks que custam \$2.50 custarem \$3.50, e faça com que todos os drinks que custam \$ 3.50 passem a custar \$ 4.50.

.....  
.....  
.....  
.....



Aponte seu lápis  
Solução

Use INSERT e DELETE para mudar a tabela drink\_info como pedido. Então, desenhe a tabela alterada ao lado.

### drink\_info

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

Mude as calorias de Kiss on the Lips para 170.

INSERT INTO drink\_info VALUES ('Kiss on the Lips', 5.5, 42.5, 'roxo', 'S', 170);

DELETE FROM drink\_info WHERE calorias = 171;

Essim que  
A sua pode  
realmente

Mude os valores do amarelo para dourado.

INSERT INTO drink\_info VALUES ('Blackthorn', 3, 8.4, 'dourado', 'S', 33),

('Greyhound', 4, 14, 'dourado', 'S', 50);

DELETE FROM drink\_info WHERE cor = 'amarelo';

drink\_info

name_do_drink	preço	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	dourado	s	33
Blue Moon	2.5	3.2	azul	s	12
Oh My Gosh	3.5	8.6	laranja	s	35
Lime Fizz	3.5	5.4	verde	s	24
Kiss on the Lips	5.5	42.5	roxo	s	170
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	s	17
Greyhound	4	14	dourado	s	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	s	80
Soda and It	3.8	4.7	vermelho	N	19

Assim que sua tabela deveria se parecer, após ter feito as mudanças.  
Mas pode estar em uma ordem diferente, mas lembre-se, a ordem  
realmente não quer dizer nada.

Esta é mais uma de suas pegadinhas?

Esta não é uma pegadinha, mas é uma questão em  
que você precisa pensar bastante. Se trocar  
os drinks de \$2.50 por \$3.50 e os de  
\$3.50 para \$4.50, você terá aumentado  
o valor do Blue Moon em dois dólares. Ao invés  
disso, precisaria alterar os valores mais altos  
primeiro (\$3.50 para \$4.50), e só então o  
Blue Moon de \$2.50 para \$3.50.



Faça com que todos os drinks que custam \$2.50 custarem \$3.50, e faça  
com que todos os drinks que custam \$3.50 passem a custar \$4.50.

`INSERT INTO drink_info VALUES ('Oh My Gosh', 4.5, 8.6, 'laranja', '5', 35);`

`DELETE FROM drink_info WHERE preco = 3.5;`

`INSERT INTO drink_info VALUES ('Blue Moon', 3.5, 3.2, 'azul', '5', 12);`

`('Lime Fizz', 3.5, 5.4, 'verde', '4', 24);`

`DELETE FROM drink_info WHERE preco = 2.5;`

Pontos extras se você colocou os dois comandos `INSERT` em um só!

## Cuidado com o seu DELETE

Cada vez que apagar os registros, você corre o risco de acidentalmente apagar registros que não tinha a intenção de apagar. Use como exemplo se tivéssemos que adicionar um novo registro para Mr. Hobo:

Aqui está a informação que precisamos adicionar, e o INSERT para fazê-lo.

Mr. Hobo visto na Tracy's.

```
INSERT INTO palhaco_info
VALUES
('Mr. Hobo', 'Tracy\'s', 'M, charuto,
cabelo preto, chapéu comprido', 'violino');
```

Use o DELETE cuidadosamente

Certifique-se que incluiu a cláusula WHERE com precisão para mirar as linhas exatas que realmente quer apagar.

DELETADO

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	
Zippo	Shopping Millstone	F, terno laranja, calças largas	cantar
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Bonzo	Parque do Dickson	M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	montando em carrinho
Mr. Hobo	Festa de Eric Gray	M, charuto, cabelo preto, chapéu comprido	violino
Mr. Hobo	Tracy's	M, charuto, cabelo preto, chapéu comprido	violino

Agora seja você o DELETE



## SEJA o DELETE

Abaixo, você vai encontrar uma série de cláusulas WHERE para um código DELETE projetado para limpar a tabela palhaco\_info da página anterior. Imagine quais códigos nos ajudam e quais criam novos problemas.

`DELETE FROM palhaco_info`

Isto nos ajuda? Se não, explique o porquê.

`WHERE visto_ultimo = 'Hospital Oakland' ;`

.....

.....

`WHERE atividades = 'violino' ;`

.....

.....

`WHERE visto_ultimo = 'Parque do Dickson'  
AND nome = 'Mr. Hobo' ;`

.....

.....

`WHERE visto_ultimo = 'Hospital Oakland'  
AND visto_ultimo = 'Parque do Dickson' ;`

.....

.....

`WHERE visto_ultimo = 'Hospital Oakland' OR  
visto_ultimo = 'Parque do Dickson' ;`

.....

.....

`WHERE nome = 'Mr. Hobo' OR visto_ultimo =  
'Hospital Oakland' ;`

.....

.....

Agora escreva um único comando DELETE que limpe todos os dados extras dos registros de Mr. Hobo sem tocar em nenhum dos outros.

.....

.....

.....



## Solução do "SEJA o DELETE"

Abaixo, você vai encontrar uma série de cláusulas WHERE para um código DELETE projetado para limpar a tabela palhaco\_info na página anterior. Imagine quais códigos nos ajudam e quais criam novos problemas.

```
DELETE FROM palhaco_info
```

✓ Scooter também tem um registro que se iguala a este.  
WHERE visto\_ultimo = 'Hospital Oakland';

✗ Não queremos apagar o registro novo.  
WHERE atividades = 'violino';

WHERE visto\_ultimo = 'Parque do Dickson'  
AND nome = 'Mr. Hobo';

✗ O AND significa que ambos devem ser verdadeiros.  
WHERE visto\_ultimo = 'Hospital Oakland'  
AND visto\_ultimo = 'Parque do Dickson';

WHERE visto\_ultimo = 'Hospital Oakland' OR  
visto\_ultimo = 'Parque do Dickson';

WHERE nome = 'Mr. Hobo' OR visto\_ultimo =  
'Hospital Oakland';

Agora escreva um único comando DELETE que  
limpe todos os dados extras dos registros de Mr.  
Hobo sem tocar em nenhum dos outros.

Isto nos ajuda? Se não, explique o porquê.

Deleta apenas um registro do Mr. Hobo.  
Também deleta registro do Scooter.

Deleta todos os registros de Mr. Hobo,  
inclusive os novos.

Apaga apenas um dos  
registros antigos

Não deleta nada.

Deleta os registros de Bonzo e Scooter juntos  
com os registros antigos de Mr. Hobo

Deleta todos os registros do Mr. Hobo, incluindo  
registro novo e deleta o registro do Scooter.

DELETE FROM palhaco\_info  
WHERE nome = 'Mr. Hobo'  
AND visto\_ultimo <> 'Tracy's';



Parece que você apagou coisas que não tinha a intenção. Talvez pudesse usar o **SELECT** primeiro para verificar o que irá deletar se utilizar está cláusula **WHERE** em particular.

**Correto! A não ser que esteja absolutamente certo de que sua cláusula WHERE deletará as linhas que desejar, você deveria utilizar o SELECT primeiro para ter certeza.**

Já que ambos podem utilizar as mesmas cláusulas WHERE, as linhas que o comando **SELECT** exibe vão atingir nas linhas que você deletará, se usar esta mesma cláusula WHERE.

É um método seguro de ter certeza que não está apagando nada acidentalmente e irá ajudá-lo a ter certeza de que está alcançando todos os dados que deseja deletar.

## O problema com DELETE impreciso

**DELETE** é ardiloso. Se não tomar cuidado, os dados errados serão o alvo. Vamos podemos evitar mirar os dados errados se adicionarmos mais um passo ao nosso programa de dois passos **INSERT-DELETE**.

Aqui está um programa de TRÊS PASSOS que você pode seguir:

Altere apenas os registros que deseja usando o comando **SELECT** primeiro.

- 1 Primeiro, **SELECT** (selecione) os registros que sabe que serão removidos para confirmar que irá deletar apenas os registros planejados e nenhum registro que não deve ser apagado.

**SELECT FROM palhaco\_info  
WHERE  
atividades = 'dançar';**

nome	visto_ultimo	aparencia	atividades
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar

- 2 Próximo, **INSERT** (insira) o novo registro

**INSERT INTO palhaco\_info  
VALUES  
( 'Zippo' , 'Shopping Millstone' , 'F, terno laranja, calças largas' , 'dançar, cantar' );**

*INSERT (insira) o registro usando todos os dados originais apenas alterando a coluna que precisa alterar.*

nome	visto_ultimo	aparencia	atividades
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar

- 3** Finalmente, DELETE o registro antigo com a mesma cláusula WHERE que você usou com o comando SELECT no começo deste programa de três passos.

```
DELETE FROM palhaco_info
WHERE
atividades = 'dançar';
```

← Utilize a cláusula WHERE que utilizou para SELECT (selecionar) o registro no passo 1 para achar e deletar o registro antigo.

nome	visto_ultimo	aparencia	atividades
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar

Agora, nos restou apenas o registro novo.

nome	visto_ultimo	aparencia	atividades
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar



Não seria um sonho se eu pudesse alterar um registro em apenas um passo sem me preocupar se meu novo registro foi apagado junto com o antigo, mas sei que isto é apenas uma utopia.

## Altere seus dados com UPDATE (Atualizar)

Até o momento, você já deve estar confortável utilizando INSERT e DELETE para manter suas tabelas atualizadas. Nós até observamos algumas formas de usá-las juntas para, indiretamente, modificar uma linha em particular. Entretanto, ao invés de inserir uma nova linha e deletar a antiga, você pode reutilizar uma linha que já está na tabela, alterando apenas os valores das colunas que quer alterar. O comando SQL é chamado de UPDATE, e ele faz exatamente o que ele significa, atualizar. Ele atualiza uma coluna, ou colunas, para novos valores. E da mesma forma que o SELECT e o DELETE, você pode atribuir uma cláusula WHERE para indicar que linha deseja UPDATE (atualizar).

Aqui está o comando UPDATE em ação:

```
UPDATE notas_donuts
    → SET
        Aqui é onde digemos qual
        deverá ser o novo valor.
    tipo = 'cobertura'
WHERE tipo = 'cobertura simples';
```

Arqui está uma cláusula WHERE padrão, da mesma forma que como viu com SELECT e DELETE.

A palavra-chave SET diz ao Sistema SQL que ele precisa alterar a coluna antes do sinal de igual para conter o valor depois do sinal de igual. No caso acima, estamos alterando 'cobertura simples' para somente 'cobertura' na nossa tabela. O WHERE diz para mudar somente linhas onde está digitado 'cobertura simples'.

### **notas\_donuts**

empresa	horario	data	tipo	nota	comentarios
Krispy King	8:50	27/09	cobertura simples	10	quase perfeito
Duncan's Donuts	8:59	25/08	NULL	6	gorduroso
Starbuzz Coffee	19:35	24/05	bolo de canela	5	Não são fresquinhos, mas são gostosos
Duncan's Donuts	19:03	26/04	geléia	7	Sem geléia suficiente



### **notas\_donuts**

empresa	horario	data	tipo	nota	comentarios
Krispy King	8:50	27/09	cobertura	10	quase perfeito
Duncan's Donuts	8:59	25/08	NULL	6	gorduroso
Starbuzz Coffee	19:35	24/05	bolo de canela	5	Não são fresquinhos, mas são gostosos
Duncan's Donuts	19:03	26/04	geléia	7	Sem geléia suficiente

## Regras do UPDATE

- Você pode utilizar o UPDATE para alterar uma única coluna ou uma porção delas. Adicione mais pares da coluna = valor para a cláusula SET e ponha uma vírgula após cada uma delas:

```
UPDATE sua_tabela
    SET primeira_coluna = 'novo valor'
    segunda_coluna = 'outro_valor';
```

- Você pode utilizar o UPDATE para atualizar uma única linha ou múltiplas linhas dependendo da sua cláusula WHERE.

---

 não existem  
 Perguntas Idiotas

UPDAT

 Usando o co  
 ultimo do re  
 Parque do D

P: O que acontece se eu não usar a cláusula WHERE?

R: Então todos os valores constantes na cláusula SET serão alterados na sua tabela.

P: Há dois sinais de igual lá na query SQL na página anterior que parecem estar fazendo coisas diferentes. Está correto?

R: Exatamente. O sinal de igual na cláusula SET diz "defina esta coluna igual a este valor", enquanto aquela na cláusula WHERE está testando para ver se o valor da coluna é igual ao valor depois do sinal de igual.

P: Eu poderia ter utilizado esta instrução para fazer a mesma coisa feita lá em cima?

```
UPDATE nota_donuts SET tipo =
    'cobertura' WHERE empresa = 'Krispy King';
```

R: Sim você pode. Isto iria atualizar a mesma coluna da mesma forma. E assim está ótimo para nossa tabela de quatro linhas. Se tivesse usado esse comando em uma tabela com centenas ou milhares de registros, você teria alterado o tipo em cada linha, contendo Krispy King.

P: Ai ai ai! Como posso me certificar de que estou atualizando apenas o que preciso?

R: Da mesma forma que viu com DELETE, a não ser que tenha certeza que está direcionando para as linhas corretas com sua cláusula WHERE, faça um SELECT primeiro!

P: Pode-se ter mais que uma cláusula SET?

R: Não, mas não deverá precisar. Você pode colocar todas as colunas e seus novos valores para elas na mesma cláusula SET, como mostramos anteriormente.

## UPDATE é o novo INSERT-DELETE

Quando você usa o UPDATE, não está apagando nada. Ao invés disso, está **reciclando o registro antigo em um novo local**.

Comece com UPDATE...

SET determina as mudanças que você fará no registro.

UPDATE nome\_tabela

SET nome\_coluna = novo valor

WHERE nome\_coluna = valor qualquer;

Nossa confiável cláusula WHERE está aqui para ajudar a direcionar precisamente qual registro queremos alterar.

Vamos vê-lo em ação como um comando que funcionará com a tabela palhaco\_info.

UPDATE (atualiza) um registro na tabela palhaco\_info.

Altera o valor na coluna visto\_ultimo para Tracy's.

UPDATE palhaco\_info

SET visto\_ultimo = 'Tracy\'s'

WHERE nome = 'Mr. Hobo'

Aqui está a cláusula WHERE para determinar com precisão o registro a ser alterado - neste caso, o registro do Mr. Hobo com a informação de visto\_ultimo sendo o valor Parque do Dickson.

Códigos UPDATE podem substituir a combinação DELETE/INSERT.

Não esqueça de inverter para essa sua aspa.

## UPDATE em ação

Isando o comando UPDATE, a coluna visto\_ultimo do registro do Mr. Hobo é alterada de Parque do Dickson para Tracy's.

Mr. Hobo visto na Tracys

Aqui está a informação que precisamos adicionar, e o comando UPDATE que utilizaremos para isso.

```
UPDATE palhaco_info
SET visto_ultimo = 'Tracy\'s'
WHERE nome = 'Mr. Hobo'
AND visto_ultimo = 'Festa de Eric Gray';
```

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Marabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
	Shopping Millstone	F, terno laranja, calças largas	dançar, cantar
	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
			cantar, dançar
			corneta, guarda-chuva
			cantar, dançar
			malabarismo, carrinho
Mr. Hobo	Tracy's ;)	hat	violino

UPDATE

Ao usar o UPDATE, você está editando in loco, então não há nenhum risco de deletar dados incorretamente (embora sobrescreva dados existentes).



Aponte seu lápis

## Atualizando os movimentos dos palhaços

Desta vez, vamos fazer direito. Preencha um comando UPDATE para cada visualização. Nós já fizemos um para você iniciar. Daí, preencha a tabela palhaco\_info na forma como ela vai ficar depois de executarmos todos os comandos UPDATE.

Zippo foi visto cantando

UPDATE palhaco\_info

SET atividades = 'cantar'

WHERE nome = 'Zippo';

Snuggles agora está vestindo  
calças largas azuis

Bonzo visto no Parque  
do Dickson

Snuffles visto montando  
em um carrinho

Mr. Hobo visto pela última  
vez na festa de Eric Gray

name	Elsie
	Pickle
	Snugg
	Mr. 1
	Clara
	Scoot
	Zipp
	Babe
	Bong
	Snif

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles			
Mr. Hobo			
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo			
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo			
Sniffles			



Aponte seu lápis  
Solução

## Atualizando os movimentos dos palhaços

Desta vez, vamos fazer direito. Preencha um comando UPDATE para cada visualização. Nós já fizemos um para você iniciar. Daí, preencha a tabela palhaco\_info na forma como ela vai ficar depois de executarmos todos os comandos UPDATE.

Zippo foi visto cantando

Snuggles agora está vestindo  
calças largas azuis

Bonzo visto no Parque do  
Dickson

Sniffles visto montando em  
um carrinho

Mr. Hobo visto pela última  
vez na festa de Eric Gray

UPDATE palhaco\_info

SET atividades = 'cantar'

WHERE nome = 'Zippo';

Nos não queremos dispensar outras informações que já estão na coluna. Certifique-se que incluído aqui.

UPDATE palhaco\_info

SET aparência = 'F, camiseta amarela, calças largas azuis.'

WHERE nome = 'Snuggles';

UPDATE palhaco\_info

SET visto\_último = 'Parque do Dickson'

WHERE nome = 'Bonzo';

UPDATE palhaco\_info

SET atividades = 'montar em carrinho'

WHERE nome = 'Sniffles';

UPDATE palhaco\_info

SET visto\_último = 'Festa de Eric Gray'

WHERE nome = 'Mr. Hobo';

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Circo BG	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	dançar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo		M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno verde e roxo, nariz pontudo	

Os outros registros não foram alterados.

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas <u>azuis</u>	corneta, guarda-chuva
Mr. Hobo	Festa de Eric Gray	M, charuto, cabelo preto, chapéu comprido	violino
Clarabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar, dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	cantar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo	Parque do Dickson	M, vestido comprido de bolinhas	cantar, dançar
Sniffles	Tracy's	M, terno roxo e verde, nariz pontudo	montar em carrinho

Apenas as partes de cada registro que SET (definimos) no UPDATE foram alterados. Finalmente, preenchemos todas as lacunas de volta na página 99.

## UPDATE (atualize) seus preços

Lembra-se de quando tentamos alterar somente os preços nos drinks da tabela `drink_info`? Nós queríamos alterar os drinks de \$2.50 para \$3.50, e os de \$3.50 para \$4.50.

`drink_info`

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	verde	S	24
Kiss on the Lips	5.5	42.5	roxo	S	171
Hot Gold	3.2	32.1	laranja	N	135
Lone Tree	3.6	4.2	vermelho	S	17
Greyhound	4	14	amarelo	S	50
Indian Summer	2.8	7.2	marrom	N	30
Bull Frog	2.6	21.5	cor de canela	S	80
Soda and It	3.8	4.7	vermelho	N	19

Vamos dar uma olhada em como podemos abordar este problema usando o comando `UPDATE` para ir a cada registro individualmente e escrever uma série de comandos `UPDATE` como este:

```
UPDATE drink_info
SET preço = 3.5 ← Preço com já com #/ acrescentado.
WHERE nome_do_drink = 'Blue Moon' ;
```



↑  
Utilizamos uma cláusula `WHERE`  
para escolher uma coluna  
específica para assim sabermos  
qual registro vamos atualizar.

 Aponte seu lápis —

Escreva comandos UPDATE para cada registro na tabela `info_drinks` para adicionar outro dólar no preço de cada um.

nome_do_drink	preco	carboidratos	cor	gelo	calorias
Blackthorn	3	8.4	amarelo	S	33
Blue Moon	2.5	3.2	azul	S	12
Oh My Gosh	3.5	8.6	laranja	S	35
Lime Fizz	2.5	5.4	green	Y	24
Kiss on the	5.5	12.5		Y	171
Hot Gold				N	135
Lone Tree					



Espere um minuto. Por que está nos dando todo este trabalho? Não existe um operador que possamos utilizar com o UPDATE ao invés de mudarmos cada registro manualmente?

### Você está certa.

Parece que alguns operadores espertos seriam justamente o que ajudaria aqui. Vamos UPDATE (atualizar) todos os preços dos drinks sem ter que fazer um por um, manualmente, arriscando sobreescriver dados que já havíamos alterado.

# Tudo o que precisamos é um UPDATE

Nossa coluna extensa é um número. Em SQL, podemos fazer operações matemáticas básicas em colunas numéricas. No caso de nossa coluna preço, podemos adicionar 1 dólar para cada linha que precisamos alterar dentro da nossa tabela. Aqui vai como:

```
UPDATE drink_info          ↙
SET preco = preco + 1;
WHERE
nome_do_drink= 'Blue Moon'
OR
nome_do_drink= 'Oh My Gosh'
OR
nome_do_drink= 'Lime Fizz' ;
```

Adicionar 1 para cada um dos 3 preços (drinks de \$2.50 e de \$3.50) que precisamos mudar.

---

## não existem Perguntas Idiotas

---

P: Posso usar subtração com um valor numérico?  
Que mais posso usar?

R: Multiplicação, divisão, subtração – você pode usar qualquer uma delas. E você pode executar estas operações utilizando outros valores numéricos, não apenas 1.

P: Você pode me dar um exemplo de quando poderei utilizar multiplicação?

R: O sinal de igual na cláusula SET diz “defina esta coluna. Suponha que tenha uma lista de itens em uma tabela, cada um com um preço. Você poderia utilizar o comando UPDATE e multiplicar o preço de cada item por um número fixo para computar o preço do produto incluindo os impostos.

P: Eu poderia ter utilizado outros operadores para fazer a mesma coisa feita lá em cima?

R: Sim, há um bom número delas. Posteriormente, falaremos sobre algumas coisas que você pode fazer com suas variáveis de texto junta as numéricas.

P: Como o quê, por exemplo? Dê-nos uma pista?

R: Ok, uma coisa que você pode fazer é usar a função UPPER() para alterar o texto da coluna inteira para letras maiúsculas. E pode até adivinhar, LOWER() fará todo o texto ficar com letras minúsculas.

P: Pode-se ter mais que uma cláusula SET?

R: Não, mas não deverá precisar. Você pode colocar todas as colunas e seus novos valores para elas na mesma cláusula SET, como mostrado acima.

Comandos UPDATE podem ser utilizados para múltiplos registros na sua tabela. Use-o com operadores básicos de matemática para manipular valores numéricos.



Acho que é bom saber como atualizar meus dados, mas em primeiro lugar, eu realmente queria poder entender como projetar melhor a tabela.

## Os dados mudam, então é crucial saber como atualizá-los.

Mas quanto melhor for seu trabalho projetando sua tabela, menos update terá que fazer no final das contas. Bons projetos de tabela o deixam livre para se concentrar nos dados na tabela.

Interessado? A seguir, daremos uma olhada de perto e, sem aflição, em projetos de tabela sob suspeita...



## Sua caixa de ferramenta SQL

O capítulo 3 em breve será somente uma lembrança, mas aqui vai um rápido lembrete dos novos comandos SQL que você já aprendeu. Para uma lista completa de dicas de ferramentas no livro, veja o Apêndice iii.

### DELETE

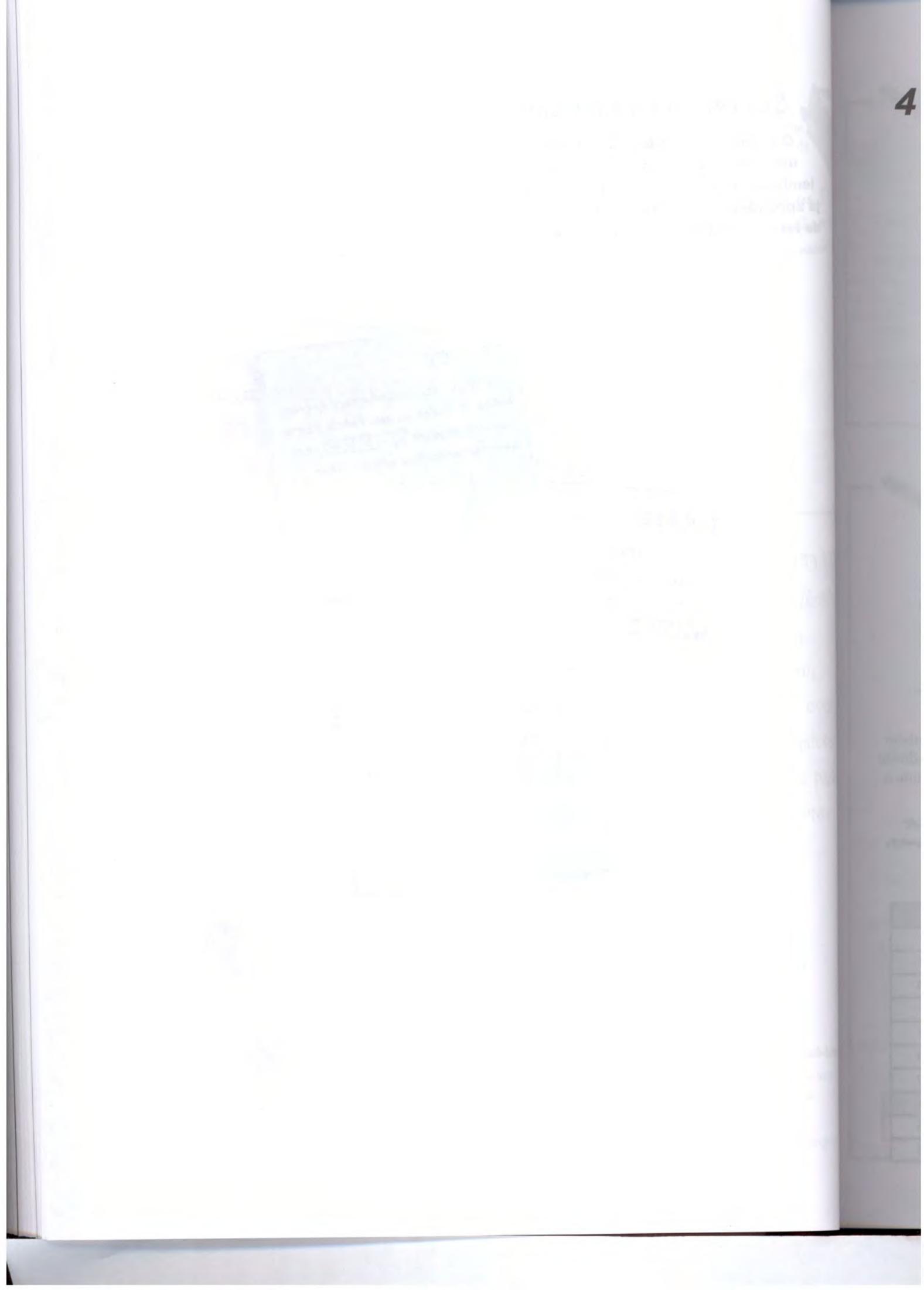
Esta é sua ferramenta para apagar linhas de dados da sua tabela. Use-a com uma cláusula **WHERE** para apontar precisamente as linhas que quer remover.

### UPDATE

O comando atualiza uma coluna existente ou colunas com um novo valor. Ele também utiliza cláusula **WHERE**.

### SET

Esta palavra-chave pertence a um comando **UPDATE** e é usada para alterar o valor de uma coluna existente.

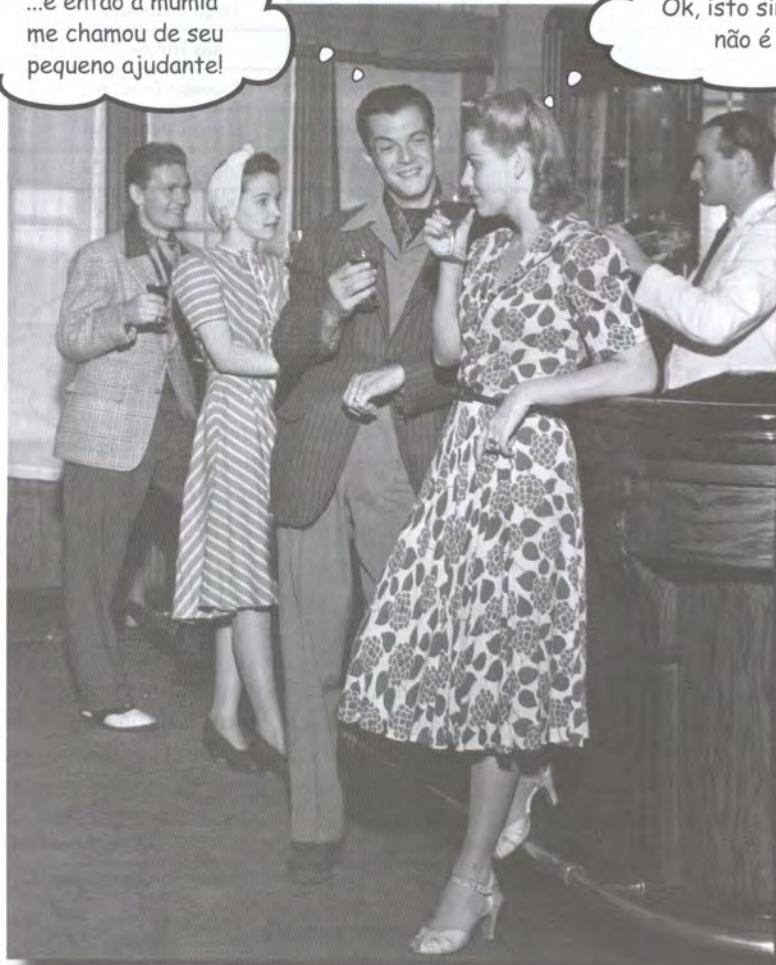


## 4 Projetos de tabela inteligentes

### Por que ser normal?

...e então a múmia  
me chamou de seu  
pequeno ajudante!

Ok, isto simplesmente  
não é normal.



**Você tem criado tabelas sem dar muita atenção a elas.** Tudo bem quanto a isso, elas funcionam. Você pode usar o SELECT, INSERT, DELETE e UPDATE nelas. Mas, à medida que você **insere mais dados**, começa a ver coisas que vai desejar ter feito para que a sua cláusula WHERE fosse mais simples. Você precisa é fazer suas tabelas *mais normais*.

## Duas tabelas de pescadores

Jack e Mark criaram uma tabela para armazenar informações sobre quebra de recordes de peixes. A tabela de Mark tem colunas para as espécies e nomes populares dos peixes e onde eles foram pescados. Ela não inclui os nomes das pessoas que pescaram o peixe.

*Esta tabela tem apenas quatro colunas. Compare com a tabela peixe\_recordes ali.*

peixe\_info

popular	especie	local	peso
robalo	M. salmoides	Lago Montgomery, GA	22 lb 4 oz
olho de peixe	S. vitreus	Lago Old Hickory, TN	25 lb 0 oz
truta, cutthroat	O. Clarki	Lago Pyramid, NV	41 lb 0 oz
perca	P. Flavescens	Bordentown, NJ	4 lb 3 oz
guelras azuis	L. Macrochirus	Lago Ketona, AL	4 lb 12 oz
peixe agulha	L. Osseus	Rio Trinity, TX	50 lb 5 oz
crappie	P. annularis	Represa Enid, MS	5 lb 3 oz
lúcio	E. americanus	Lago Dewart, IN	1 lb 0 oz
peixe dourado	C. auratus	Lago Hodges, CA	6 lb 10 oz
salmão	O. Tshawytscha	Rio Kenai, AK	97 lb 4 oz



*Eu sou um ictiólogo. Só pesquiso na minha tabela pelo nome da espécie e nome popular para obter peso e local onde ocorreu o recorde.*

Mark

A tabela de Jack contém o nome popular e o peso do peixe, mas também contém o primeiro nome e o sobrenome da pessoa que o pescou, e divide a coluna local em uma coluna contendo o nome do corpo hídrico onde o peixe foi pego e uma coluna separada para o estado.

*Esta tabela também se refere à quebra de recordes, mas tem quase o dobro de colunas.*

peixe\_recordes

primeiro_nome	sobrenome	popular	local	estado	peso	data
George	Perry	robalo	Lago Montgomery	GA	22 lb 4 oz	2/6/1932
Mabry	Harper	olho de peixe	Lago Old Hickory	TN	25 lb 0 oz	2/8/1960
John	Skimmerhorn	truta, cutthroat	Lago Pyramid	NV	41 lb 0 oz	1/12/1925
C.C.	Abbot	perca	Bordentown	NJ	4 lb 3 oz	1/5/1865
T.S.	Hudson	guelras azuis	Lago Ketona	AL	4 lb 12 oz	9/4/1950
Townsend	Miller	peixe agulha	Rio Trinity	TX	50 lb 5 oz	30/7/1954
Fred	Bright	crappie	Represa Enid	MS	5 lb 3 oz	31/7/1957
Mike	Berg	lúcio	Lago Dewart	IN	1 lb 0 oz	9/6/1990
Florentino	Abena	peixe dourado	Lago Hodges	CA	6 lb 10 oz	17/4/1996
Les	Anderson	salmão	Rio Kenai	AK	97 lb 4 oz	17/5/1985



Aponte seu lápis

Escreva uma consulta para **cada tabela** para encontrar todos os registros de New Jersey (NJ).

Sou escritor da Carretilha e Companhia.  
Preciso saber o nome dos pescadores, datas  
e locais das grandes pescas.



Aponte seu lápis

Solução

Escreva uma consulta para cada tabela para encontrar todos os registros de New Jersey (NJ).

Nós temos que usar um `LIKE` para obter os resultados da cidade e estado juntos.

Quase nunca preciso pesquisar através do estado. Eu insiro os dados com o estado na mesma coluna que a cidade.

`SELECT * FROM peixe_info`

`WHERE local LIKE '%NJ';`

popular	especie	local	peso
perca	P. Flavescens	Bordentown, NJ	4 lb 3 oz



também  
1, e divide  
o onde o

ira de  
colunas.

a
932
960
925
865
950
954
957
990
996
985

esta consulta procura diretamente na coluna estado.

Constantemente tenho que pesquisar na minha tabela pelo estado, então eu os coloquei separados na coluna para estado quando criei minha tabela.

`SELECT * FROM peixe_records`

`WHERE estado = 'NJ';`

primeiro_nome	sobrenome	popular	local	estado	peso	data
C.C.	Abbot	perca	Bordentown	NJ	4 lb 3 oz	1/5/1865



## não existem Perguntas Idiotas

**P:** Então a tabela do Jack é melhor que a do Mark?

**R:** Não. Elas são tabelas diferentes para propósitos diferentes. Mark raramente vai precisar pesquisar diretamente por estado porque ele só realmente se importa com os nomes das espécies, seus respectivos nomes populares e o peixe recorde e seu peso.

Jack, por outro lado, vai precisar fazer consulta pelo estado quando estiver fazendo uma consulta em seus dados. É por isso que a tabela dele tem uma coluna separada: para permitir que indique precisamente os estados em suas pesquisas.

**P:** Devemos evitar o LIKE quando estivermos fazendo consulta em nossas tabelas? Há algo de errado com isso?

**R:** Não há nada de errado com LIKE, mas pode ser difícil se usar nas suas consultas, e você se arrisca de obter resultados que não quer. Se suas colunas contêm informações complicadas, LIKE não é específico o suficiente para apontar os dados precisamente.

**P:** Por que consultas curtas são melhores que as longas?

**R:** Quanto mais simples a consulta, melhor. À medida que seu banco de dados cresce, e à medida que você adiciona novas tabelas, suas consultas vão se tornando mais complicadas. Se começar com a consulta mais fácil possível agora, você vai agradecer depois.

**P:** Então você está dizendo que eu sempre devo ter pequenos bits de dados nas minhas colunas?

**R:** Não necessariamente. Como você está começando a perceber entre as tabelas de Mark e Jack, depende em como usará seus dados.

Por exemplo, imagine uma tabela listando os carros para um mecânico e uma listando carros para um vendedor. O mecânico pode precisar de informações mais precisas sobre cada carro, mas uma garagem de carros precisa somente da marca do carro, modelo, e número do chassi.

**P:** Suponha que temos um endereço de nação, mas nós não temos uma coluna para o endereço, e outras colunas que separam estes dados?

**R:** Enquanto duplicar seus dados parece uma boa idéia pra você agora, leve em consideração quanto espaço vai ocupar no seu HD quando seu banco de dados alcançar proporções enormes. Cada vez que duplicar seus dados, é importante lembrar de adicionar quando se fizerem alterações.

Vamos olhar mais de perto em como projetar tabelas na melhor forma possível para você.

**Como você vai utilizar seus dados afetará em como você monta sua tabela.**



### PODER DO CÉREBRO

SQL é a linguagem utilizada por banco de dados relacionais. O que você acha que relacionais quer dizer em um banco de dados SQL?

## Uma tabela tem tudo a ver com relacionamentos

SQL é conhecida como **sistema de gestão de bancos de dados relacionais, ou RDBMS**.

Não se incomode em memorizá-lo. Nós nos importamos mesmo apenas com a palavra RELATIONAL\*.

O que tudo quer dizer é que para projetar uma tabela fera, você precisa levar em conta **como as colunas vão se relacionar** umas com as outras **para descrever algo**.

Qual é o objeto principal que você quer que seja o assunto principal da tabela?

**1. Escolha o objeto principal, aquele que você quer que sua tabela descreva.**

Como você utiliza esta tabela?

**2. Faça uma lista das informações que você precisa saber sobre seu único objeto principal quando estiver utilizando a tabela.**

Qual a maneira mais fácil de fazer a consulta destas tabelas?

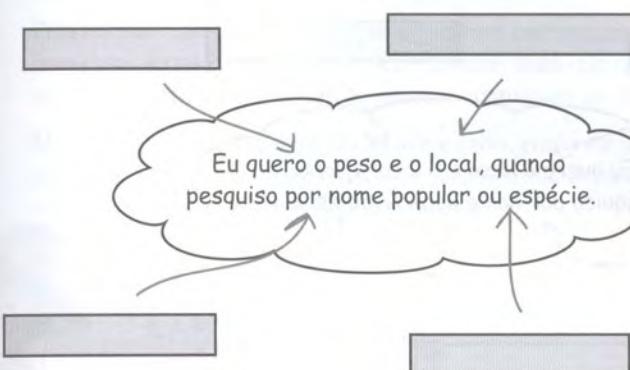
**3. Usando a lista das informações, divida a informação sobre aquele objeto principal em pedaços que possa utilizar para organizar sua tabela.**

\* Algumas pessoas acham que RELATIONAL significa muitas tabelas relacionadas entre si. Isto está errado.

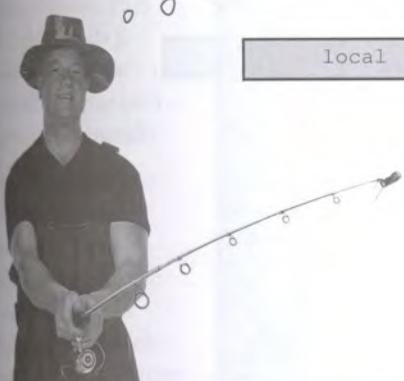
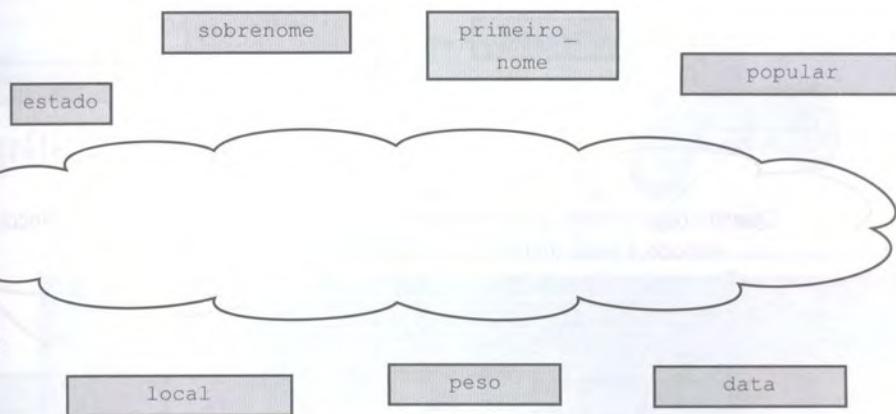


Você pode destacar as colunas que Mark, o ictiólogo, usou para descrever como ele quer selecionar a partir de sua tabela? Preencha a coluna\_principal

### Exercícios



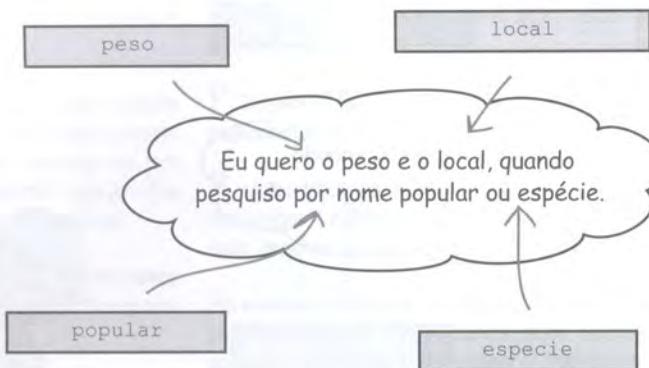
Sua vez. O escritor da Revista Carretilha e Companhia, que utiliza esta tabela para selecionar detalhes para seus artigos. Após, desenhe flechas partindo de cada coluna para onde ela se refere no comando.



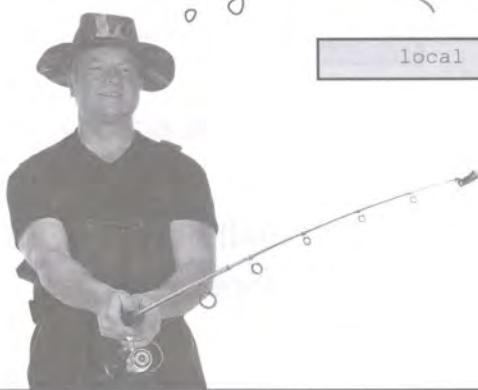
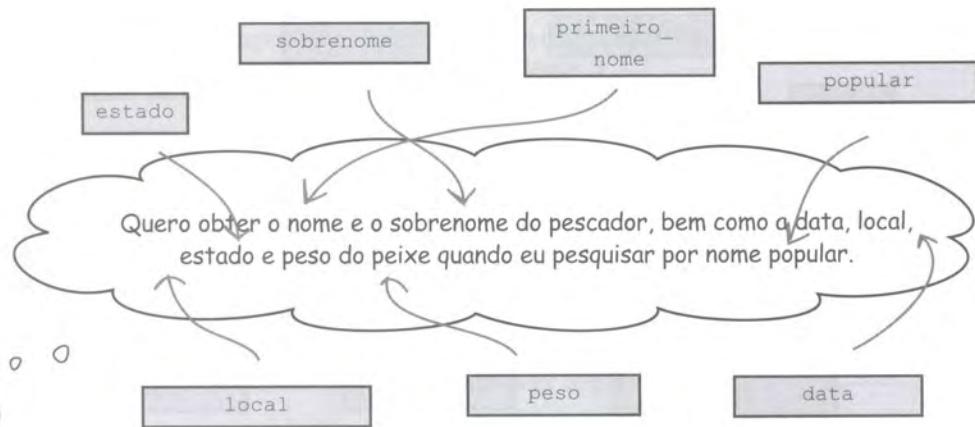


## Solução dos Exercícios

Você pode destacar as colunas que Mark, o ictiólogo, usou para descrever como ele quer selecionar a partir de sua tabela? Preencha a coluna\_principal



Sua vez. O escritor da Revista Carretilha e Companhia, que utiliza desta tabela para selecionar detalhes para seus artigos. Após, desenhe flechas partindo de cada coluna para onde ela se refere no comando.



## Dados atômicos

O que é um átomo? É aquilo que não pode ser dividido. Aqui, os dados são átomos quando se referem a seus dados. Quer dizer que ele já é o menor bloco de dados que pode ser dividido.

## 30 minutos

Imagine um exemplo de consulta que não é atômica. Por exemplo, para pedir uma pizza. Para chegar ao resultado, a consulta precisa, ele só pode dividir a consulta em uma só coluna. Isso significa que seus propósitos de busca não são atômicos. Ele precisa olhar a consulta em seu todo e não só o número da rua.

De fato, se seu resultado fossem divididos em partes menores, a consulta não seria mais útil. Isso significa que suas consultas precisam ser mais longas e mais complexas, o que pode demorar mais tempo para obter o resultado da pizza ao vizinho.



Mas por que parar por aí com a tabela de Jack? Você não poderia dividir a data em dia, mês e ano? Você poderia ainda dividir o local em nome da rua e número.

### Nós poderíamos, mas não precisamos dos dados divididos neste nível.

Pelo menos, não neste caso. Se Jack estivesse escrevendo um artigo sobre os melhores locais para ir nas suas férias para pegar um peixe grande, *então* ele poderia querer o nome da rua e o número para que os leitores pudessem encontrar acomodações em locais próximos.

Mas Jack precisava somente do local e estado, então ele adicionou tantas colunas quanto ele precisava para economizar espaço na tabela. Neste ponto, ele achou que sua tabela já estava suficientemente dividida - ela é **atômica**.



O que você acha que a palavra atômica significa em termos de dados SQL?

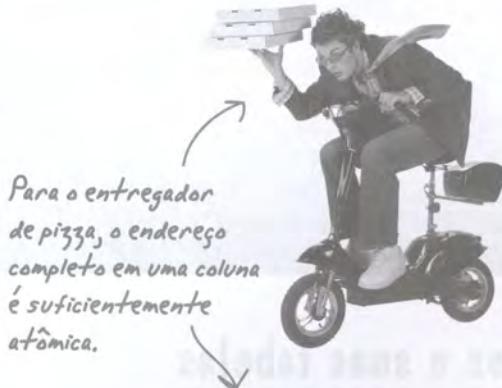
## Dados atômicos

O que é um átomo? Um pequeno pedaço de informação que não pode ou não deveria ser dividido. É o mesmo para seus dados. Quando eles são **ATÔMICOS**, isto quer dizer que ele já foi dividido até o **menor pedaço de dados que não pode ou não deve ser dividido**.

### 30 minutos ou é grátis

Imagine um entregador de pizza. Para chegar aonde precisa, ele só precisa do número da rua e endereço em uma só coluna. Para seus propósitos, ela já é atômica. Ele nunca precisa olhar apenas para o número da rua separado.

De fato, se seus dados fossem divididos em nome da rua e número, suas consultas teriam que ser mais longas e mais complicadas, fazendo ele demorar mais para entregar a pizza ao vizinho.



```

File Edit Window Help SimplePizzaFactory
+-----+
| pedido | endereco |
+-----+
| 246    | 59 N. Ajax Rapids
| 247    | 849 SQL Street
| 248    | 2348 E. PMP Plaza
| 249    | 1978 HTML Heights
| 250    | 24 S. Servlets Springs
| 251    | 807 Infinite Circle
| 252    | 32 Design Patterns Plaza
| 253    | 9208 S. Java Ranch
| 254    | 4653 W. EJB Estate
| 255    | 8678 OOA&D Orchard
+-----+
> SELECT endereco FROM pizza_deliveries WHERE pedido = 252;
+-----+
| endereco |
+-----+
| 32 Design Patterns Plaza |
+-----+
1 row in set (0.04 sec)
  
```

## Local, local, local

Agora imagine um corretor de imóveis. Ele pode querer ter uma coluna separada para o número da rua. Ele pode querer consultar apenas pelo número da rua fornecido para ver todas as casas à venda. Para ele o nome e o número da rua são cada uma, atômica.

*Para o corretor de imóveis separar o nome da rua do número permite que ele veja todas as casas à venda em uma rua específica, com uma consulta simples.*



```

File Edit Window Help IWantMyCommission
+-----+
| número | nome_rua           | tipo_propriedade | preço |
+-----+
| 59     | N. Ajax Rapids       | condomínio        | 189000 |
| 849    | SQL Street            | apartamento       | 109000 |
| 2348   | E. PMP Plaza          | casa               | 355000 |
| 1978   | HTML Heights           | apartamento       | 134000 |
| 24     | S. Servlets Springs   | casa               | 355000 |
| 807    | Infinite Circle        | condomínio        | 143900 |
| 32     | Design Patterns Plaza  | casa               | 465000 |
| 9208   | S. Java Ranch          | casa               | 699000 |
| 4653   | SQL Street             | apartamento       | 115000 |
| 8678   | OOA&D Orchard          | casa               | 355000 |
+-----+
> SELECT preço, tipo_propriedade FROM imobiliária WHERE nome_rua = 'SQL Street';
+-----+
| preço      | tipo_propriedade |
+-----+
| 109000.00  | apartamento      |
| 115000.00  | apartamento      |
+-----+
2 rows in set (0.01 sec)

```

## Dados atômicos e suas tabelas

Há algumas perguntas que você tem que fazer para descobrir o que precisa colocar em suas tabelas:



**1. Qual o Objeto principal que sua tabela descreve?**

*Sua tabela descreve palhaços, vacas, donuts, pessoas?*



**2. Como você usará a tabela para chegar até este objeto?**

*Projete sua tabela para uma fácil consulta.*



**3. As suas colunas possuem dados atômicos para fazer consultas pequenas e diretas ao ponto?**

## não existem Perguntas Idiotas

P: O átomo não é bem pequeno? Eu não deveria dividir meus dados em pedaços bem pequenos, então?

R: Não. Fazer seus dados atômicos significa dividi-los em menores pedaços que precisar para criar uma tabela eficiente, não só os menores pedaços possíveis de se conseguir.

Não divida sua tabela mais do que precisa, se não precisar de colunas extras, não as adicione sem motivo.

P: Como os dados atômicos me ajudam?

R: Eles ajudam a assegurar se que os dados na sua tabela permaneçam corretos. Por exemplo, se você tem uma coluna para o número das ruas, pode se certificar que apenas números possam ser inseridos naquela coluna.

Dados atômicos também permitem que você execute consultas mais eficientes porque as consultas ficam mais fáceis de serem escritas e gastam menos tempo para serem executadas; o que conta muito quando se tem uma quantidade de dados massiva armazenadas.



Aponte seu lápis

Aqui estão as regras oficiais dos dados atômicos. Para cada regra, rasgue duas tabelas hipotéticas que violam cada regra.

**REGRA 1:** Uma coluna com dados atômicos não pode conter muitos valores do mesmo tipo de dados naquela coluna.

*A coluna interesses da tabela meus\_contatos de Greg viola esta regra.*

**REGRA 2:** Uma tabela com dados atômicos não pode conter múltiplas colunas com o mesmo tipo de dados.

*A tabela drinks\_faceis viola esta regra.*

to?



## Aponte seu lápis

### Solução

Aqui estão as regras oficiais dos dados atômicos. Para cada regra, rascunhe duas tabelas hipotéticas que violem cada regra.

**REGRA 1:** Uma coluna com dados atômicos não pode ter muitos valores do mesmo tipo de dados naquela coluna.

*É claro que suas respostas serão diferentes, mas aqui vai um exemplo:*

nome_comida	ingredientes
pão	farinha, leite, ovo, fermento, óleo
salada	alface, tomate, pepino

Ainda lembra da tabela do Greg? Ela tem uma coluna para hobbies que geralmente contém múltiplos interesses, fazendo da consulta pesadelo.

É a mesma coisa aqui: imagine tentar encontrar tomate entre todos estes outros ingredientes.

**REGRA 2:** Uma tabela com dados atômicos não pode ter múltiplas colunas com o mesmo tipo de dados.

professor	estudante1	estudante2	estudante3
Mr. Martini	Joe	Ron	Kelly
Mr. Howard	Sanjaya	Tim	Julie

*Têm muitas colunas para estudar*



Agora que você sabe as regras oficiais e os três passos para produzir os dados atômicos, dê uma olhada em cada tabela anterior deste livro e explique o porquê de cada uma ser ou não atômica.

Tabela do Greg, página 32 .....

Tabela de notas dos donuts, página 65 .....

Tabela de Palhaco\_info, página 101 .....

Tabela dos drinks, página 61 .....

Tabela Informativa dos Peixes, página 132 .....

## Razões para ser normal

Quando sua empresa de consultoria decolar e você precisar contratar mais designers de dados SQL, não seria ótimo se não precisasse gastar horas explicando como suas tabelas funcionam?

Além de fazer suas tabelas do jeito NORMAL quer dizer que elas seguem algumas regras básicas que seus novos designers iriam entender. E a boa notícia é que nossas tabelas com dados atômicos já são a metade do caminho.

Fazer dados atômicos é o primeiro passo para criar uma tabela NORMAL.



## Solução dos Exercícios

Tabela do Greg, página 32 *Não é atômica. A coluna "interesse" e "procura" violam a regra 1.*

Tabela de notas dos donuts, página 65 *Atômica. Ao contrário da tabela drinks\_faceis, cada coluna comporta um tipo diferente de informação. É ao contrário da coluna "atividades" da tabela dos palhaços, cada coluna tem apenas um pedaço de informação nela.*

Tabela de notas dos donuts, página 101 *Não é atômica. A coluna "atividades" tem mais que uma atividade em alguns registros, portanto viola a regra 1.*

Tabela dos drinks, página 61 *Não é atômica. Há mais de uma coluna "quantidade", o que viola a regra 2.*

Tabela Informativa dos Peixes, página 132 *Atômica. Cada coluna comporta um tipo diferente de informação e cada coluna tem apenas um pedaço de informação nela.*

Palhaç

Ainda lembr  
nossa antiga  
contém mui

Tom
Elsi
Pickl
Snugg
Mr. H
Clarab
Scoo
Zipp
Bab
Bon
Sniff

## O benefício de tabelas normais

1. Tabelas normais não terão dados duplicados, o que reduzirá o tamanho de seu banco de dados.

Evitar dados duplicados vai poupar espaço no disco.

2. Com menos dados para pesquisar, suas consultas serão mais rápidas.





Minhas tabelas não  
são tão grandes. Por que  
eu deveria me preocupar em  
deixá-las normais?

### Porque mesmo quando suas tabelas são pequeninas elas melhoram.

E tabelas crescem. Se você começar com  
uma **tabela normalizada**, não terá  
que voltar atrás e alterar sua tabela quando  
suas consultas começarem a ficar lentas.

## Palhaços não são normais

Lembra da tabela dos palhaços? Rastrear palhaços tornou-se uma mania nacional, e essa antiga tabela não vai ter sucesso porque as colunas aparência e atividades contêm muitos dados. Para nossos propósitos, esta tabela não é atômica.

*Estas duas colunas são  
realmente difíceis de consultar  
porque contêm muitos dados!*

### palhaco\_info

nome	visto_ultimo	aparencia	atividades
Elsie	Centro da Terceira Idade Cherry Hill	F, cabelo vermelho, vestido verde, pés enormes	balões, carrinho
Pickles	Festa de Jack Green	M, cabelo laranja, terno azul, pés enormes	mímica
Snuggles	Mercado da Bola	F, camiseta amarela, calças largas e vermelhas	corneta, guarda-chuva
Mr. Hobo	Festa do Eric Gray	M, charuto, cabelo preto, chapéu comprido	violino
Clorabelle	Centro da Terceira Idade Belmont	F, cabelo rosa, flores gigantes, vestido azul	gritar e dançar
Scooter	Hospital Oakland	M, cabelo azul, terno vermelho, nariz enorme	balões
Zippo	Shopping Millstone	F, terno laranja, calças largas	cantar
Babe	Autopeças do Earl	F, toda rosa e brilhosa	malabarismo, carrinho
Bonzo	Parque do Dickson	M, vestido comprido de bolinhas	cantar, dançar
Snuffles	Tracy's	M, terno verde e roxo, nariz pontudo	montando em um carrinho

Aponte seu lápis -

Vamos tornar a tabela de palhaços mais atômica, assumindo que você precisa pesquisar seus dados entre as colunas **aparência** e **atividades**, como também na coluna **visto\_ultimo**, escreva algumas escolhas melhores para colunas.

## Metade do caminho para 1NF

Lembre-se, nossa tabela está apenas metade do caminho para ser normal quanto ela tiver com os dados atômicos. Quando ela estiver completamente normal, estaremos no PRIMEIRA FORMA NORMAL ou 1FN.

Para ser 1NF, a tabela deve seguir essas duas regras:

Nós já sabemos como fazer isto.

→ Cada linha de dados deve conter dados atômicos.

Para fazer nossa tabela completamente normal, precisamos estabelecer para cada registro uma chave primária.

Cada linha de dados deve ter um identificador único, conhecido como Chave Primária.



### PODER DO CÉREBRO

Que tipo de colunas você acha que seriam boas Chaves Primárias?

## Regras da CHAVE PRIMÁRIA

A coluna na sua tabela que será sua chave primária deve ser designada para isso no momento em que criar sua tabela. Em poucas páginas, criaremos uma tabela e designar uma chave primária, mas antes disso, vamos olhar de perto no que vem a ser uma chave primária.



### A chave primária é utilizada exclusivamente para identificar cada registro

O que significa que os dados na chave primária não podem ser repetidos. Imagine uma tabela com as colunas mostradas abaixo. Você acha que alguma delas seria uma boa chave primária?

CPF	sobrenome	primeiro_nome	telefone
-----	-----------	---------------	----------

Tendo em vista que o CPF é atribuído unicamente a uma só pessoa, talvez ele pudesse ser uma chave primária.

Todas estas três colunas podem conter valores duplicados - por exemplo, você provavelmente tem mais de um registro para alguém chamado Pedro, ou múltiplas pessoas que vivem juntas e compartilham o mesmo número de telefone, então provavelmente estas não são boas opções para serem uma chave primária.



Uma chave primária é uma coluna na sua tabela que torna cada registro único.



### Veja Isto!

Tome cuidado ao utilizar CPF como sua chave primária para seus registros.

Com o roubo de identidades só aumentando, as pessoas não querem fornecer o número do CPF - e com uma boa razão. Ele é importante demais para se arriscar. Você pode garantir absolutamente que seu banco de dados é seguro? Se não é, todos esses números de CPF podem ser roubados, bem como a identidade de seus clientes.



### A chave primária não pode ser NULL

Se ela é NULL não pode ser única porque outros registros também podem ser NULL.

### A chave primária deve ser atribuída no momento em que o registro é inserido.

Quando inserir um registro sem uma chave primária, você corre o risco de acabar com uma chave primária NULL e linhas duplicadas em sua tabela, o que viola a primeira forma normal.



### A chave primária deve ser compacta

Uma chave primária deveria conter apenas as informações necessárias para ser única e nada mais.



### Os valores da chave primária não podem ser alterados.

Se pudesse alterar os valores da sua chave, você estaria se arriscando, acidentalmente, a defini-la como um valor já utilizado. Lembre-se que ela deve permanecer única.



### PODER DO CÉREBRO

Dadas todas estas regras, você consegue pensar em uma boa chave primária para usar em uma tabela?

Olhe para as tabelas anteriores contidas neste livro. Alguma delas têm uma coluna que contenha verdadeiramente valores únicos?

Espere, se não devo utilizar CPF como chave primária, mas ainda precisa ser compacto, não ser NULL, ser inalterável, o que eu deveria utilizar?



### A melhor chave primária pode ser uma nova chave primária.

Quando o assunto é criar chaves primárias, seu melhor palpite seria criar uma coluna que contenha um número único. Pense em uma tabela com informações sobre pessoas, mas com uma coluna adicional contendo um número. No exemplo abaixo, vamos chamá-la de ID.

Se não fosse pela coluna ID, os registros de John Brown seriam idênticos. Mas neste caso, eles são na verdade duas pessoas diferentes. A coluna ID faz de cada registro uma exclusividade. Esta tabela está em Primeira Forma Normal.

id	sobrenome	primeiro_nome	apelido
1	Brown	John	John
2	Ellsworth	Kim	Kim
3	Brown	John	John
4	Petrillo	Maria	Maria
5	Franken	Esme	Em

← um registro para John Brown

← Também um registro para John Brown, mas a coluna ID mostra que ele é um John Brown diferente do primeiro.



## Dica Nerd

Há um grande debate no mundo SQL sobre usar chaves primárias sintéticas, ou inventadas (como a coluna ID acima) ao invés de utilizar chaves naturais – dados que já estão na tabela (como o chassi de um carro ou o CPF de alguém). Não iremos escolher um lado, mas iremos falar sobre chave primária com mais detalhe no capítulo 7.

## não existem Perguntas Ídiotas

**P:** Você disse "Primeira" Forma Normal. Isto quer dizer que existe a Segunda Forma Normal? Ou terceira?

**R:** Sim, de fato existe a Segunda e a Terceira Forma Normal, cada uma delas aderindo a um conjunto de regras cada vez mais rígido. Abordaremos a Segunda e a Terceira Forma Normal no capítulo 7.

**P:** Então nós já transformamos nossas tabelas para terem valores atômicos. Alguma delas já está em 1FN?

**R:** Não. Até agora, nenhuma tabela sequer que tenhamos criado tem uma chave primária, um valor único.

**P:** A coluna comentários na tabela de donuts não me parece atômica. Não há nenhuma forma razoável de se consultar aquela tabela com facilidade.

**R:** Você está absolutamente correto. Aquele campo em particular não é atômico, mas o projeto daquela nossa tabela não exigia que ele fosse. Se quiséssemos restringir os comentários a apenas um conjunto de palavras pré-determinadas, aquele campo poderia ser atômico, mas desta forma não conteria comentários verdadeiros e espontâneos.

## Ficando NORMAL

Está na hora de voltar atrás e normalizar nossas tabelas. Precisamos tornar nossos dados atômicos e adicionar as chaves primárias. Criar uma chave primária é normalmente algo que fazemos ao utilizar o código CREATE TABLE.



Você se lembra como adicionar colunas a uma tabela existente?

## Consertando a tabela do Greg

A partir do que você já viu até agora, é assim que teria que consertar a tabela do Greg:

Consertar a tabela do Greg – passo 1: `SELECT`(selecionar) todos os dados e salvá-los de alguma forma.

Consertar a tabela do Greg – passo 2: Criar uma nova tabela normal.

Consertar a tabela do Greg – passo 3: `INSERT`(inserir) todos os dados antigos na tabela nova, alterando cada linha para adequar-se à nova estrutura.

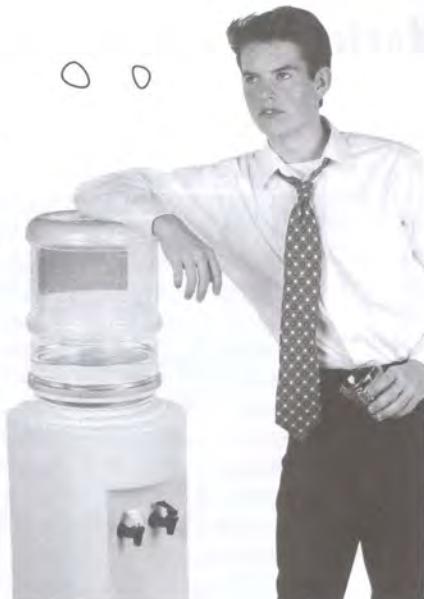
Agora você pode deixar de lado a sua tabela antiga.

Espere um segundo. Eu já tenho uma tabela cheia de dados. Você não pode estar realmente esperando que eu utilize o comando DROP TABLE como fiz no capítulo 1 e digitar todos aqueles dados novamente, apenas para criar uma chave primária para cada registro....

**Então, agora sabemos que a tabela do Greg não é perfeita.**

Elá não é atômica e não é chave primária. Mas para a sorte dele, não é necessário viver com a tabela antiga, e você **não tem que abandonar a sua antiga**.

Podemos adicionar uma chave primária na tabela do Greg e fazer as colunas se tornarem mais atômicas usando apenas um novo comando. Mas antes de tudo, vamos dar uma volta ao passado...



## O comando CREATE TABLE que nós escrevemos

Greg precisa de uma chave primária e depois de toda aquela conversa sobre dados atômicos, ele percebeu que algumas coisas que poderiam ser feitas para tornar suas colunas mais atômicas. Mas, antes de darmos como consertar uma tabela já existente, vamos primeiramente verificar como criariammos a tabela.

Aqui está a tabela que criamos lá atrás no Capítulo 1.

### CREATE TABLE meus\_contatos

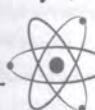
(

sobrenome VARCHAR (30),  
 primeiro\_nome VARCHAR (20),  
 email VARCHAR (50),  
 aniversario DATE,  
 profissao VARCHAR (50),  
 local VARCHAR (50),  
 estado\_civil VARCHAR (20),  
 interesses VARCHAR (100),  
 procura VARCHAR(100)

);

Não há chave  
primária aqui.

Estas colunas poderiam ter  
sido mais atômicas quando  
criamos a tabela?



### PODER DO CÉREBRO

Mas e se você não tem seu comando CREATE TABLE impresso em algum lugar? Você sabe alguma forma de acessá-lo?

# tabela

## Mostre-me o dinheiro

E se você utilizar o comando `DESCRIBE meus_contatos` para verificar o código que você utilizou quando criou a tabela? Você verá algo mais ou menos assim:

Column	Type	Null	Key	Default	Extra
sobrenome	varchar(30)	YES		NULL	
primeiro_nome	varchar(20)	YES		NULL	
email	varchar(50)	YES		NULL	
sexo	char(1)	YES		NULL	
aniversario	date	YES		NULL	
profissao	varchar(50)	YES		NULL	
local	varchar(50)	YES		NULL	
estado_civil	varchar(20)	YES		NULL	
interesses	varchar(100)	YES		NULL	
procura	varchar(100)	YES		NULL	

Mas o que nós realmente queremos ver é o código `CREATE` aqui e não os campos da tabela para que possamos imaginar o que teríamos feito lá no início sem ter que escrever o comando `CREATE` todo novamente.

O comando `SHOW CREATE_TABLE` irá retornar o comando `CREATE TABLE` que pode recriar sua tabela, com exceção dos dados nela inseridos. Desta forma, você sempre pode ver como a tabela que está olhando pode ser criada. Experimente:

```
SHOW CREATE TABLE meus_contatos;
```

## Comando para economia de tempo

Dê uma olhada no comando que utilizamos para criar a tabela, e o código abaixo que o comando `SHOW CREATE_TABLE meus_contatos` lhe deu. Eles não são idênticos, mas se colar o texto abaixo no seu comando `CREATE TABLE`, o resultado final será o mesmo. Você não precisa remover os acentos graves ou configurações de dados, mas fica mais caprichado se o fizer.

```
CREATE TABLE `meus_contatos` (
  `sobrenome` varchar(30) default NULL,
  `primeiro_nome` varchar(20) default NULL,
  `email` varchar(50) default NULL,
  `sexo` char (1) default NULL,
  `aniversario` date default NULL,
  `profissao` varchar(50) default NULL,
  `local` varchar(50) default NULL,
  `estado_civil` varchar(20) default NULL,
  `interesses` varchar(100) default NULL,
  `procura` varchar(100) default NULL,
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Os sinais abrindo e fechando os nomes das colunas e o nome da tabela são chamados de acentos graves. Eles aparecem quando executamos o comando `SHOW CREATE TABLE`.

A não ser que digamos ao Sistema SQL que é uma boa prática especificar a coluna com valores não nulos quando cria a tabela.

Você não precisa se preocupar quanto à última linha do texto após fechar os parênteses. Ela especifica como as linhas serão armazenadas e qual o conjunto de caracteres a ser utilizado. A definição padrão está ótima, por enquanto.

A não ser que tenha deletado a tabela original, você terá que dar um nome diferente.

O CRE  
Aqui está o grava e a ú NULL, e ao chave prim

Criamos u coluna ch id\_contat comporta integer g a chave p para nos Cada valo tabela se tornando nossa te

P: Então você pode ser NULL duplicada?

R: Basicamente (insere) valores na coluna id\_contacto, exemplo, o primeiro id\_contacto será 2, etc.

Embora pudesse deixar o código mais caprichado (removendo a última linha e os acentos graves), você pode apenas copiar e colar para o comando create table.

## CREATE TABLE com uma CHAVE PRIMÁRIA

Ali está o comando que o SHOW CREATE TABLE meus\_contatos meus\_contatos nos deu. Nós removemos os acentos e a última linha. No topo da lista de colunas adicionamos uma coluna id\_contato que estamos definindo como NOT NULL, e ao final da lista adicionamos uma linha PRIMARY KEY, que definimos para usar nossa coluna id\_contato como a chave primária.

```
CREATE TABLE meus_contatos
(
    id_contato INT NOT NULL
        Lembre-se, a coluna da chave primária tem que ser NOTNULL. Se a chave primária contém um valor NULL, ou nenhum valor, você não pode garantir que ela identificara especificamente cada linha da tabela.
    sobrenome varchar (30) default NULL,
    primeiro_nome varchar (20) default NULL,
    sexo char (1) default NULL,
    email varchar(50) default NULL,
    aniversario date default NULL,
    profissao varchar(50) default NULL,
    local varchar(50) default NULL,
    estado_civil varchar(20) default NULL,
    interesses varchar(100) default NULL,
    procura varchar(100) default NULL,
    PRIMARY KEY (id_contato)
)
```

*Vamos uma nova coluna chamada id\_contato que vai importar valores integer que será a chave primária para nossa tabela. Um valor nesta tabela será único, tornando a tabela átomica a essa tabela.*

*Aqui é onde especificamos nossa chave primária. Sintaxe bem simples: nós apenas dizemos PRIMARY KEY e colocamos entre parênteses o nome da coluna que utilizamos para isso - neste caso nossa nova coluna id\_contato.*

### não existem Perguntas Idiotas

**P:** Então você diz que a PRIMARY KEY não pode ser NULL. O que mais impede ela de ser vazia?

**R:** Basicamente, você. Quando você INSERT valores na sua tabela, inserirá um valor na coluna id\_contato que é nova a cada vez. Por exemplo, o primeiro comando INSERT irá definir id\_contato em 1, o próximo id\_contato

**P:** É um problemão ter que definir um valor novo para a coluna PRIMARY KEY cada vez que vou inserir um novo registro. Não há um jeito mais fácil?

**R:** Há duas maneiras. Uma é usando uma coluna nos seus dados que você sabe que seja único como uma chave primária. Nós já mencionamos que isto é ardiloso (por exemplo, o problema de usar o número do CPF).

O jeito mais fácil é criar uma coluna inteira completamente nova apenas para armazenar um valor único, como id\_contato nesta página. Você pode dizer ao seu Sistema SQL para automaticamente inserir um número para utilizar as palavras-chave. Vire a página para mais detalhes.

**P:** Eu posso utilizar SHOW para qualquer outra coisa além do comando CREATE?

**R:** Você pode utilizar SHOW para exibir uma coluna individualmente na sua tabela.

SHOWS COLUMNS FROM nome\_tabela;  
Este comando exibirá todas as colunas na sua tabela e seus tipos de dados ao longo com quaisquer outros detalhes específicos da coluna.

SHOW CREATE DATABASE nome\_database;  
Da mesma forma que SHOW CREATE TABLE, você irá obter o comando que poderia recriar seus bancos de dados.

---

não existem  
Perguntas Idiotas

---

`SHOW INDEX FROM nome_tabela;`  
Este comando exibirá quaisquer colunas que estão indexadas e o tipo de index que elas possuem. Até agora, o único index que olhamos foram as chaves primárias, mas este comando se tornará mais útil à medida que você aprende mais.

E há mais um outro comando que é muito especial:

`SHOW WARNINGS;`

Se você conseguir uma mensagem em seu console de que seu comando SQL tenha originado avisos, digite isto para ver os avisos atuais.

Há mais uma boa porção, mas aqueles são os que estão relacionados a coisas que já fizemos anteriormente.

**P:** Então, qual é a do acento grave que aparece quando uso o `SHOW CREATE TABLE`? Você tem certeza de que não vou precisar dela?

**R:** Ele existe porque às vezes seu Sistema SQL pode não conseguir dizer que o nome da sua tabela seja o nome de fato da tabela. Se utilizar o acento grave para ficar entre os nomes das colunas, você poderá de fato (embora seja uma péssima idéia) usar uma palavra-chave reservada para o SQL como um nome de coluna.

Por exemplo, suponha que queira nomear uma coluna com o nome `select` por alguma razão muito estranha. Esta expressão não funcionaria:

`select varchar (50)`

Mas esta já funcionaria:

`'select' varchar (50)`

**P:** O que há de errado em utilizar palavras-chave de nome de colunas então?

**R:** Você é permitido fazê-lo, mas é uma idéia. Imagine o quanto confuso suas consultas tornariam, e a perturbação de digitar aquelas graves sendo que você pode se safar de tudo. Além do mais, `select` não é um nome de coluna bom, pois não diz nada sobre que tipos de dados encontram ali.

## 1, 2, 3 ... auto incrementando

Adicionar a palavra-chave `AUTO_INCREMENT` para nossa coluna `id_contato` fará com que nosso Sistema SQL automaticamente preencha aquela coluna com um valor que inicia na linha 1 com um valor de 1 e vai aumentando em acréscimos de 1.

```
CREATE TABLE meus_contatos
(
    id_contato INT NOT NULL AUTO_INCREMENT,
    sobrenome varchar (30) default NULL,
    primeiro_nome varchar (20) default NULL,
    email varchar(50) default NULL,
    sexo char (1) default NULL,
    aniversario date default NULL,
    profissao varchar(50) default NULL,
    local varchar(50) default NULL,
    estado_civil varchar(20) default NULL,
    interesses varchar(100) default NULL,
    procura varchar(100) default NULL,
    PRIMARY KEY (id_contato)
)
```

Isto é tudo. Apenas adicione a palavra-chave `AUTO_INCREMENT` caso esteja utilizando a maioria das espécies de SQL (usuários de MySQL e MySQLi devem ignorar este alerta, a palavra-chave é `INDEX` juntamente com um valor inicial e valor de acréscimo. Chegue à referência SQL para informação específica.)

A palavra-chave faz basicamente o que você espera que ela faça: ela começa do 1 e vai aumentando toda vez que inserir uma nova linha.



Ok, parece simples e suficiente, mas como uso um comando `INSERT` se aquela coluna já será preenchida por mim? Eu posso acidentalmente sobrepor o valor já inserido naquela já existente?

### O que você acha que vai acontecer?

Melhor ainda, experimente você mesma e veja o que acontece.



have como  
a péssima  
nsultas se  
es acentos  
utilizá-las.  
coluna muito  
s de dados

re  
do a  
15 SQL  
junto  
egue sua  
>  
espera  
em /,

so  
er  
á

ie  
e

- 1 Escreva um comando CREATE TABLE abaixo para armazenar nome e sobre nome de pessoas. Sua tabela deverá ter uma coluna chave primária com AUTO\_INCREMENT e outras duas colunas atômicas.

.....  
.....  
.....

- 2 Abra seu terminal SQL ou interface Gráfica e execute seu comando CREATE TABLE.

- 3 Experimente cada comando INSERT abaixo. Circule aqueles que funcionam.

```
INSERT INTO sua_tabela (id, primeiro_nome, sobrenome)
VALUES (NULL, 'Marcia', 'Brady');
```

```
INSERT INTO sua_tabela (id, primeiro_nome, sobrenome)
VALUES (1, 'Jan', 'Brady');
```

```
INSERT INTO sua_tabela
VALUES ('', 'Bobby', 'Brady');
```

```
INSERT INTO sua_tabela (primeiro_nome, sobrenome)
VALUES ('Cindy', 'Brady');
```

```
INSERT INTO sua_tabela (id, primeiro_nome, sobrenome)
VALUES (99, 'Peter', 'Brady');
```

- 4 Todos os nomes Brady conseguiram? Rascunha sua tabela e seu conteúdo depois de tentar os comandos INSERT acima.

sua_tabela		
<i>id</i>	<i>primeiro_nome</i>	<i>sobrenome</i>



## Solução dos Exercícios

- 1 Escreva um comando CREATE TABLE abaixo para armazenar nome e sobre nome de pessoas. Sua tabela deverá ter uma coluna chave primária com AUTO\_INCREMENT e outras duas colunas atômicas.

```
CREATE TABLE sua_tabela
(
    id INT NOT NULL AUTO_INCREMENT,
    primeiro_nome VARCHAR(20),
    sobrenome VARCHAR(30),
    PRIMARY KEY (id)
);
```

- 2 Abra seu terminal SQL ou interface GUI e execute seu comando CREATE TABLE.

- 3 Experimente cada comando INSERT abaixo. Circule aqueles que funcionam.

*(Circulado)*

```
INSERT INTO sua_tabela (id, primeiro_nome, sobrenome)
VALUES (NULL, 'Marcia', 'Brady');
```

```
INSERT INTO sua_tabela (id, primeiro_nome, sobrenome)
VALUES (1, 'Jan', 'Brady');
```

*(Circulado)*

```
INSERT INTO sua_tabela
VALUES ('', 'Bobby', 'Brady');
```

*(Circulado)*

```
INSERT INTO sua_tabela (primeiro_nome, sobrenome)
VALUES ('Cindy', 'Brady');
```

*(Circulado)*

```
INSERT INTO sua_tabela (id, primeiro_nome, sobrenome)
VALUES (99, 'Peter', 'Brady');
```

*Este último comando 4 funciona, mas ele sobrescreve o valor da coluna AUTO-INCREMENT*

- 4 Todos os nomes Brady conseguiram? Rascunha sua tabela e seu conteúdo depois de tentar os comandos INSERT acima.

sua_tabela		
id	primeiro_nome	sobrenome
1	Marcia	Brady
2	Bobby	Brady
3	Cindy	Brady
99	Peter	Brady

*Parece que perdemos Jan porque tentamos dar a ela um índice que estava determinado para Marcia, Marcia, Marcia!*

## Adiciona

Aqui está o código para meus\_contatos

ALTER

ADD CO

ADD PR

ADD COLU  
dig - adiciona  
como id\_cri

## Perguntas Idiotas

**P:** Por que a primeira consulta, aquela com NULL para a coluna id, inseriu a linha quando a coluna id é NOT NULL?

**R:** Mesmo que pareça que ele não deveria ser bem-sucedido, o AUTO\_INCREMENT simplesmente ignora o NULL. No entanto, se a coluna não fosse do tipo AUTO\_INCREMENT, você obteria um erro e a linha não seria inserida. Experimente.



Olhe, você não está me acalmando. Claro, eu posso colar o código a partir do SHOW CREATE TABLE, mas ainda estou sentindo que terei que apagar minha tabela pela segunda vez e começar a inserir todos aqueles registros novamente, apenas para inserir a coluna chave primária.

**Você não terá que recomeçar; ao invés disso, pode utilizar um comando ALTER.**

Uma tabela com dados não precisa ser deixada de lado, nem excluída, nem recriada. Nós podemos na verdade alterar a tabela existente. Mas para fazer isso, iremos emprestar o comando ALTER e algumas de suas palavras-chave do Capítulo 5.

## Adicionando uma CHAVE PRIMÁRIA a uma tabela existente

ésta é o código para adicionar uma chave primária AUTO\_INCREMENT para a tabela meus\_contatos do Greg.

Aqui está nosso novo comando SQL, ALTER.

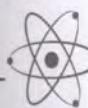
Aqui está o código para adicionar a nova coluna à tabela. Parece familiar, não?

FIRST diz ao software para fazer com que a nova coluna seja a primeira na lista. Ele é opcional, mas é uma boa maneira de inserir sua chave primária em primeiro.

```
ALTER TABLE meus_contatos
ADD COLUMN id_contato INT NOT NULL AUTO_INCREMENT FIRST,
ADD PRIMARY KEY (id_contato);
```

O COLUMN faz justamente o que ela faz: adicionar uma coluna à tabela e nomeá-la como id\_contato.

Você deve reconhecer a linha que atribui a chave primária.



### PODER DO CÉREBRO

Você acha que com isso os valores serão adicionados à nova coluna para aqueles registros que já se encontram na tabela ou somente para os próximos registros inseridos? Como você pode checar?

# ALTER TABLE e adicionar uma PRIMARY KEY

Tente o código você mesmo. Abra seu terminal SQL. Use o banco de dados gregs\_list e digite o comando:

```
File Edit Window Help Alterations
> ALTER TABLE meus_contatos
-> ADD COLUMN id_contato INT NOT NULL AUTO_INCREMENT FIRST,
-> ADD PRIMARY KEY (id_contato);

Query OK, 50 rows affected (0.04 sec)
Records: 50 Duplicates: 0 Warnings: 0
```

Isto diz que  
foi adicionada a  
coluna para os  
50 registros já  
existentes em nossa  
tabela. Você não terá  
tantos assim.

Isto é bem esperto! Eu tenho uma chave  
primária completa com valores. O ALTER TABLE  
pode me ajudar a inserir uma coluna para número  
de telefone?

Para ver o que aconteceu com sua tabela, tente o  
comando `SELECT * from meus_contatos;`

A coluna `id_contato` foi  
adicionada primeiro na  
tabela antes de todas  
as outras colunas.

Por utilizarmos  
`AUTO_INCREMENT`,  
a coluna foi preenchida  
assim que cada  
registro na tabela foi  
atualizado.

<code>id_contato</code>	<code>sobrenome</code>	<code>primeiro_nome</code>	<code>email</code>
1	Anderson	Jillian	jill_anderson@yahoo.com
2	Joffe	Kevin	kj@simuduck.com
3	Newsome	Amanda	aman21uv@yahoo.com
4	Garcia	Ed	ed99@mysoftware.com
5	Roundtree	Jo-Ann	jojo@yahoo.com
6	Briggs	Chris	chris_briggs@emai...com

A próxima vez que inserirmos um novo registro,  
será dado à coluna `id_contato` um valor maior em /  
que o maior que o mais alto `id_contato` na tabela. Se  
o último registro tinha um `id_contato` de 23, o  
próximo será 24.

Lembre-se, este não é  
final da tabela; Greg  
é uma porção de contatos.

Greg conseguirá inserir sua coluna de telefone? Vá para o capítulo 5 para descobrir.



## Sua caixa de ferramenta SQL

Você já tem o capítulo 4 na palma da mão. Olhe para todas as novas ferramentas que adicionou para sua caixa de ferramentas agora! Para uma lista completa de dicas de ferramentas, veja o Apêndice iii.

### DADOS ATÔMICOS

Os dados na sua tabela são atômicos se eles estão divididos até os menores pedaços que você precisa.

### REGRA 1:

Uma coluna com dados atômicos não podem ter muitos valores do mesmo tipo de dados naquela coluna.

### REGRA 2:

Uma tabela com dados atômicos não pode ter múltiplas colunas com o mesmo tipo de dados.

### SHOW CREATE TABLE

Use este comando para ver a sintaxe correta para criar uma tabela já existente.

### PRIMARY KEY

Uma coluna ou um conjunto de colunas que identifica uma linha de dados na sua tabela de forma única.

### PRIMEIRA FORMA NORMAL (1FN)

Cada linha de dados deve conter valores atômicos, e cada linha de dados deve ter um identificador único.

### AUTO\_INCREMENT

Quando utilizada na expressão da sua coluna aquela coluna dará automaticamente um valor inteiro cada vez que um comando `INSERT` foi executado.

 Aponte seu lápis  
Solução  
da página 143

Vamos tornar a tabela de palhaços mais atômica, assumindo que você precisa pesquisar seus dados entre as colunas aparência e atividades, como também na coluna visto\_ultimo, escreva algumas escolhas melhores para colunas.

5 A

Definitivamente, não há uma resposta correta aqui.

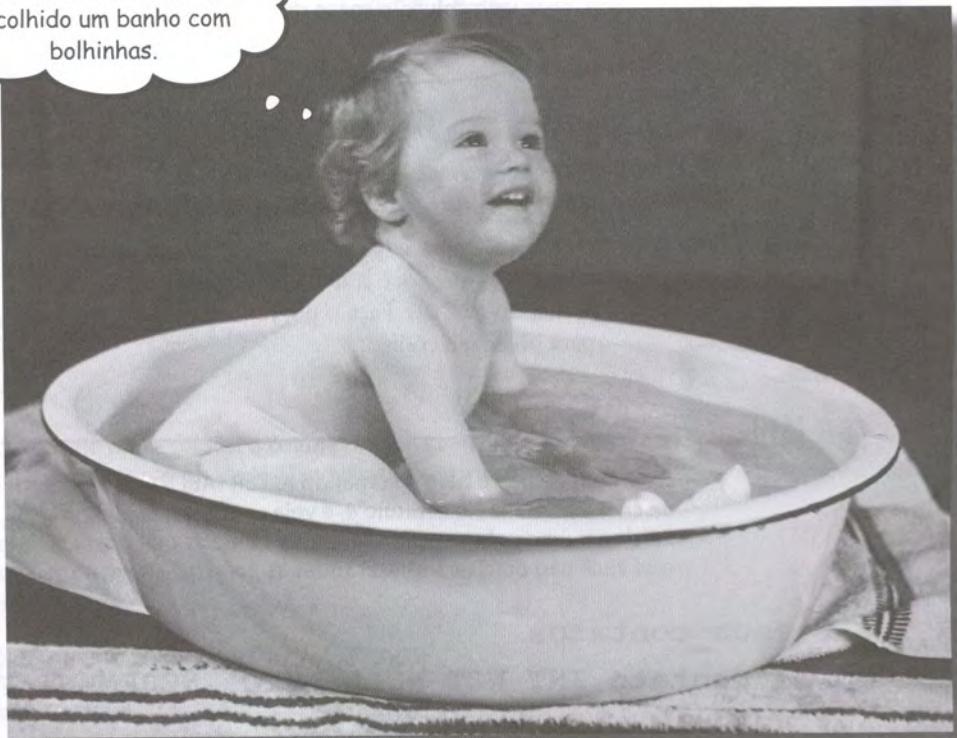
O melhor que você pode fazer é puxar informações como sexo, cor da camiseta, cor das calças, tipo do chapéu, instrumento musical, transporte, balões (sim ou não para os valores), cantar (valores sim/não).

Para tornar esta tabela atômica, você terá que pegar aquelas múltiplas atividades em colunas separadas e aquelas múltiplas aparências apresentadas separadamente.

Pontos extras se você quiser separar a coluna local em endereço, cidade e estado.

## Reescrevendo o Passado

Se eu pudesse fazer  
isso novamente, teria  
escolhido um banho com  
bolhinhas.



**Você gostaria corrigir os erros do seu passado?** Bem, agora é sua chance. Apenas usando o comando ALTER, você pode aplicar a todas as lições que tem aprendido sobre as tabelas que criou há dias, meses e até anos atrás. Melhor ainda, você pode fazer isso sem alterar seus dados. No momento que tiver passado por aqui, você saberá o que o normal realmente significa, e estará apto a aplicar isso em todas as suas tabelas, passadas ou presentes.

Você já sabe que o comando ALTER serve para mudar estruturas de tabelas (adicionar, remover, alterar colunas, etc.). Ele também serve para mudar os dados de uma tabela. Se você se interessou nisso,

# Nós precisamos fazer algumas mudanças

Greg gostaria de fazer mais algumas alterações em sua tabela, mas ele não quer perder nenhum dado.

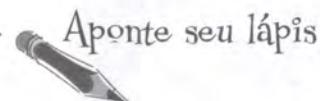
File	Edit	Window	Help	KeyedUp
id_contato	sobrenome	primeiro_nome	email	
1	Anderson	Jillian	jill_anderson@yahoo.com	
2	Joffe	Kevin	kj@simuduck.com	
3	Newsome	Amanda	aman2luv@yahoo.com	
4	Garcia	Ed	ed99@mysoftware.com	
5	Roundtree	Jo-Ann	jojo@yahoo.com	
6	Briggs	Chris	chris@myemail.com	



Então, posso adicionar uma coluna número de telefone depois de tudo?

## Sim, você pode utilizar ALTER TABLE para adicioná-la facilmente.

Na verdade, nós achamos que deveria se arriscar a fazê-la, já que você já deu uma olhada no comando ALTER. Faça o próximo exercício para obter seu código.



Olhe bem de perto para o comando ALTER TABLE que utilizamos para adicionar a coluna da chave primária no capítulo 4, e veja se consegue elaborar seu próprio código para adicionar uma coluna para número de telefone que possa comportar 10 dígitos. Observe que você não precisará utilizar todas as palavras-chave no seu novo comando.

```
ALTER TABLE meus_contatos
ADD COLUMN id_contato INT NOT NULL AUTO_INCREMENT FIRST,
ADD PRIMARY KEY (id_contato);
```

Escreva seu comando ALTER TABLE aqui:

.....

.....

.....

Você pode ainda dizer ao software onde colocar a coluna telefone com a palavra-chave AFTER. Veja se consegue pensar sobre onde colocar a palavra-chave para ADD (adicionar) a nova coluna logo após a coluna primeiro\_nome.

Escreva seu novo comando ALTER TABLE aqui:

.....

.....

.....

 Aponte seu lápis

Solução

Olhe bem de perto para o comando ALTER TABLE que utilizamos para adicionar a coluna da chave primária no capítulo 4, e veja se consegue elaborar seu próprio código para adicionar uma coluna para número de telefone que possa comportar 10 dígitos. Observe que você não precisará utilizar todas as palavras-chave no seu novo comando.

ALTER TABLE meus\_contatos

ADD COLUMN id\_contato INT NOT NULL AUTO\_INCREMENT FIRST,  
ADD PRIMARY KEY (id\_contato);

↑      ↑      ↑

As palavras-chave que deixamos de lado  
foram: NOTNULL, AUTO\_INCREMENT  
e FIRST.

Escreva seu comando ALTER TABLE aqui:

O nome da tabela que estamos  
alterando ainda é meus\_contatos.

ALTER TABLE meus\_contatos

ADD COLUMN telefone VARCHAR(10);

Aqui está o pedaço que diz  
tutamente ao comando ALTER  
que você quer alterar a tabela.

Nós assumimos que todos os nossos  
números de telefone terão 10  
caracteres de comprimento. Greg não  
pensou nos números de telefone de  
outros países.

O nome da nova  
coluna é telefone.

Você pode ainda dizer ao software onde colocar a coluna telefone com a palavra-chave AFTER. Veja se consegue pensar sobre onde colocar a palavra-chave para ADD (adicionar) a nova coluna logo após a coluna primeiro\_nome.

Escreva seu novo comando ALTER TABLE aqui:

ALTER TABLE meus\_contatos

ADD COLUMN telefone VARCHAR(10);

AFTER primeiro\_nome;

A palavra-chave AFTER (após) seguida do  
nome da coluna que quer que seja sua nova  
coluna. Ela coloca a coluna telefone logo após a  
coluna primeiro\_nome.

AFTER é opcional. Se não usá-la, a  
coluna é adicionada ao final da tabela.

Você viu que pode utilizar as palavras-chave FIRST  
(primeiro) e AFTER (após) sua\_coluna, mas pode  
também utilizar BEFORE (antes) sua\_coluna e LAST (por  
último), e também SECOND (segundo), e THIRD (terceiro),  
e você entendeu a idéia.

Por Detrás  
das Cenas





## Ímã de Geladeira - Palavras-Chave de SQL

Utilize os ímãs abaixo para alterar a posição da coluna **telefone** que foi adicionada. Crie quantos comandos puder, depois rascunhe nas colunas após ter executado o comando.

telefone	id_contato	sobrenome	primeiro_nome	email
----------	------------	-----------	---------------	-------

```
ALTER TABLE meus_contatos  
ADD COLUMN telefone VARCHAR(10)
```

id_contato	sobrenome	primeiro_nome	email	telefone
------------	-----------	---------------	-------	----------

```
ALTER TABLE meus_contatos  
ADD COLUMN telefone VARCHAR(10)
```

id_contato	telefone	sobrenome	primeiro_nome	email
------------	----------	-----------	---------------	-------

```
ALTER TABLE meus_contatos  
ADD COLUMN telefone VARCHAR(10)
```

id_contato	sobrenome	telefone	primeiro_nome	email
------------	-----------	----------	---------------	-------

```
ALTER TABLE meus_contatos  
ADD COLUMN telefone VARCHAR(10)
```

↑  
Adicione as palavras-chave ao final do comando.



Use o ponto-e-vírgula quantas vezes achar necessário.



## Ímã de Geladeira - Palavras-Chave de SQL

Utilize os ímãs abaixo para alterar a posição da coluna **telefone** que foi adicionada. Crie quantos comandos puder, depois rascunhe nas colunas após ter executado o comando.

**ALTER TABLE meus\_contatos**

**ADD COLUMN telefone VARCHAR(10)**

**FIRST**

**;**

← **FIRST**(primeiro) coloca a coluna **telefone** antes de todas as outras colunas.

telefone	id_contato	sobrenome	primeiro_nome	email
----------	------------	-----------	---------------	-------

**ALTER TABLE meus\_contatos**

**ADD COLUMN telefone VARCHAR(10)**

**LAST**

**;**

← **LAST**(último) coloca a coluna **telefone** depois de todas as outras colunas, o mesmo fará **FIFTH**(quinto) e, ainda, sem especificar qualquer posição da coluna.

**ALTER TABLE meus\_contatos**

**ADD COLUMN telefone VARCHAR(10)**

**FIFTH**

**;**

id_contato	sobrenome	primeiro_nome	email	telefone
------------	-----------	---------------	-------	----------

**ALTER TABLE meus\_contatos**

**ADD COLUMN telefone VARCHAR(10)**

**SECOND**

**;**

← **SECOND**(segundo) coloca a coluna **telefone** em segundo, o mesmo faz **BEFORE** (antes) (se utilizá-lo com a coluna **sobrenome**).

**ALTER TABLE meus\_contatos**

**ADD COLUMN telefone VARCHAR(10)**

**BEFORE**

**sobrenome**

**;**

id_contato	telefone	sobrenome	primeiro_nome	email
------------	----------	-----------	---------------	-------

**ALTER TABLE meus\_contatos**

**ADD COLUMN telefone VARCHAR(10)**

**AFTER**

**sobrenome**

**;**

← **AFTER**(após) sobrenome coloque a coluna **telefone**. Se você tivesse um ímã contendo a palavra **THIRD**, ele teria feito a mesma coisa.

id_contato	sobrenome	telefone	primeiro_nome	email
------------	-----------	----------	---------------	-------

## Alterações de tabelas

O comando ALTER permite que altere tudo na sua tabela sem precisar reinserir seus dados. Mas cuidado, se alterar os tipos de dados de uma coluna para outro tipo, estará se arriscando a perder seus dados.

## Reconst

Vamos começa tabela que prec

ste nome de colun  
nos diz nada sobr  
que contém.

# Alterações Dataville

NOSSOS SERVIÇOS PARA TABELAS EXISTENTES:

**CHANGE** (mudar) o nome e os tipos de dados de uma coluna já existente \*

**MODIFY** (modificar) os tipos de dados e a posição de uma coluna existente \*

**ADD** (adicionar) uma coluna a sua tabela – você escolhe os tipos de dados

**DROP** (eliminar) uma coluna da sua tabela \*

\* Perda de dados pode ocorrer, não oferecemos garantia.

SERVIÇOS ADICIONAIS

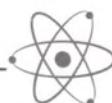
Reorganize suas colunas  
(somente disponível ao usar ADD).

É só uma pequena alteração, não vai doer nada.

Vamos utilizar coluna sendo :

## Renome

A tabela tem a uma lista de p utilizar o ALT



### Poder do Cérebro

Por que esta tabela poderia precisar de alterações?

projetus

numero	descricao do projeto	empreiteiro do projeto
1	pintar exterior da casa\	Murphy
2	remodelar a cozinha	Valdez
3	instalação de piso de madeira	Keller
4	fazer o telhado	Jackson

## Reconstrução total de tabelas

Vamos começar nossas alterações com uma tabela que precisa de alterações maiores.

Bem-vindo à Reconstrução Total de Tabelas! Nas próximas páginas iremos pegar uma tabela toda desarrumada e transformá-la em uma tabela que todo banco de dados teria orgulho em tê-la.

Este não nos diz o suficiente sobre o que esta coluna deve conter.

O nome de coluna não diz nada sobre o que ela deve conter.

projetus

Talvez possamos colocar algumas sublinhas para tornar o nome mais legível.

numero	descricao do projeto	empreiteiro do projeto
1	pintar exterior da casa	Murphy
2	remodelar a cozinha	Valdez
3	instalação de piso de madeira	Keller
4	fazer o telhado	Jackson



Enquanto o nome da tabela e os das colunas não estão bons, os dados na tabela são válidos e nós gostaríamos de mantê-los.

Vamos utilizar o comando DESCRIBE para verificar como esta tabela está construída. Ele nos mostra se há alguma coluna sendo a chave primária e quais os tipos de dados sendo inseridos em cada coluna.

```

File Edit Window Help BadTableDesign
> DESCRIBE projetus;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| Numero | int(11) | YES | | NULL | |
| Descricao do projeto | varchar(50) | YES | | NULL | |
| Empreiteiro do projeto | varchar(10) | YES | | NULL | |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

```

## Renomeando a tabela

A tabela tem alguns problemas em sua situação atual, mas graças ao ALTER, nós a faremos adequada para conter uma lista de projetos para melhoria de casas, necessárias para uma casa em particular. Nossa primeiro passo será utilizar o ALTER TABLE e dar para a nossa tabela um nome significativo.

ALTER TABLE projetus

RENAME TO lista\_projetos;

projetus é o antigo nome da nossa tabela.

Isto é praticamente inglês! Nós queremos RENOMEAR nossa tabela.

lista\_projetos é o novo nome que estamos dando para nossa tabela.



Esta descrição vai ajudá-lo a entender como precisa alterar a tabela. Encontre as colunas nesta expressão que descreva como iremos utilizar nossa tabela, aí então preencha os nomes das colunas.

**id\_proj**



Para tornar nossa tabela NORMAL, adicionaremos também uma chave primária com um número de projeto único para ele. Depois, precisaremos de colunas para descrever cada melhoramento, a data de início, custo estimado e o nome da companhia empreiteira contratada, bem como seu número de telefone.





Esta descrição vai ajudá-lo a entender como precisa alterar a tabela. Encontre as colunas nesta expressão que descreva como iremos utilizar nossa tabela, aí então preencha os nomes das colunas.

Certifique-se que nomes abreviados, como **id\_proj**, faça sentido para você e para todos que possam vir a trabalhar com o banco de dados.

Não se preocupe se o nome das suas colunas não correspondem exatamente como ilustradas aqui. Alguns abreviamentos de siglas são aceitáveis, desde que, é claro, possa ser armazenado.

**descricao\_proj**

**id\_proj**

**data\_inicio**

**custo\_est**

**empresa\_nome**

Para tornar nossa tabela NORMAL, adicionaremos também uma chave primária com um número de projeto único para ele. Depois, precisaremos de colunas para descrever cada melhoramento, a data de início, custo estimado e o nome da companhia empreiteira contratada, bem como seu número de telefone.

**empresa\_fone**

- numero
- descricao
- melhoramento
- empresa
- empresa\_fone

Isto nos deixa:

- data\_inicio

Como ela é  
coluna secundária  
e conterá a

## Nós precisamos fazer alguns planejamentos

lista\_projetos

numero	descricao do projeto	empreiteiro do projeto
1	pintar exterior da casa	Murphy
2	remodelar a cozinha	Valdez
3	instalação de piso de madeira	Keller
4	fazer o telhado	Jackson

Há que os dados para três das nossas novas colunas já estão lá. Ao invés de criarmos todas as novas colunas, podemos RENAME (renomear) nossas colunas existentes. Ao renomeá-las, que contêm dados válidos, nós não precisaremos inserir os dados para as novas colunas.



### PODER DO CÉREBRO

Qual coluna existente pode ser uma boa candidata a nossa chave primária?

## Reinstrumentalizando nossas colunas

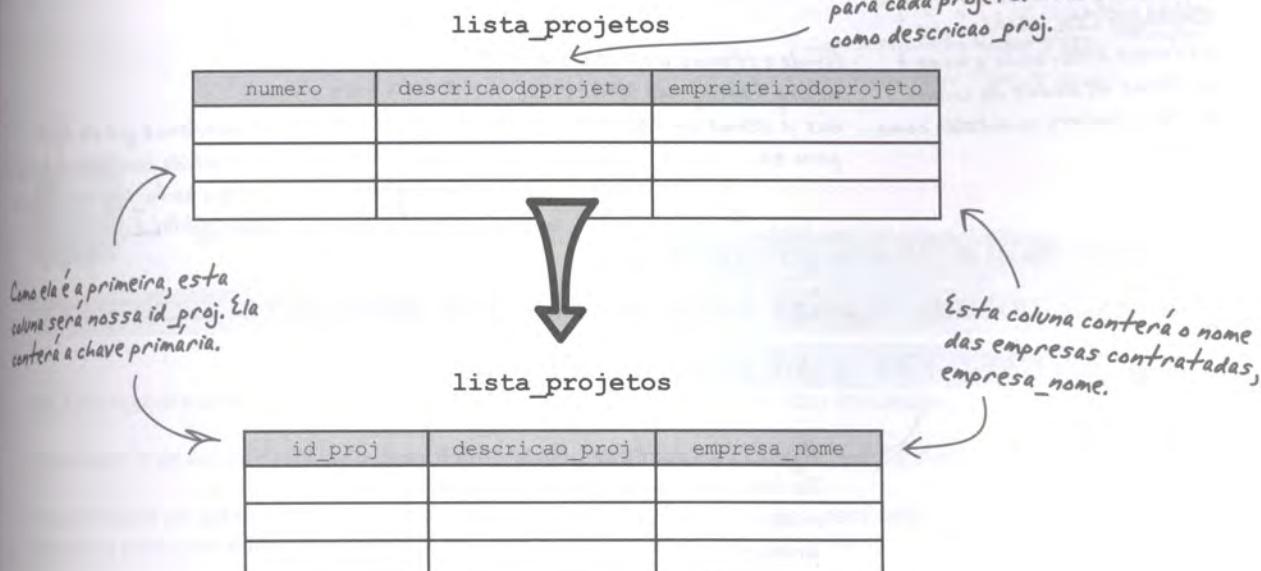
Agora temos um plano inicial e podemos ALTER (alterar) as colunas já na nossa tabela para que se encaixem com três dos nossos novos nomes de colunas:

- numero é a nossa chave primária: **id\_proj**
- descricao do projeto é uma descrição de cada projeto de melhoramento da casa: **descricao\_proj**
- empreiteiro do projeto é o nome da empresa contratada ou **empresa\_nome** para abreviar.



Nos deixa com as três colunas custo\_est, empresa\_fone e data\_inicio a serem adicionadas.

Esta coluna vai conter a descrição para cada projeto. Nós a nomearemos como descricao\_proj.



## Mudanças estruturais

Nós decidimos utilizar colunas existentes para três de nossas colunas necessárias. Além de simplesmente alterar nomes, deveríamos dar uma olhada de perto nos tipos de dados que cada uma dessas colunas armazena.

Aqui está a descrição que olhamos anteriormente.

```
File Edit Window Help BadTableDesign
> DESCRIBE projetus
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| numero     | int(11)    | YES  |     | NULL    |      |
| descricao do projeto | varchar(50) | YES  |     | NULL    |      |
| empreiteiro do projeto | varchar(10) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```



### Musculação cerebral

Olhe para cada uma das colunas TYPE e decida se os tipos de dados atuais estão adequados para os dados que iremos inserir nesta tabela.

## Altere

Iremos alterar a descrição da tabela que precisava ter uma vírgula entre

## ALTER e CHANGE

Para o nosso próximo passo, alteraremos a coluna numero para ter um novo nome, `id_proj` e defini-la como `AUTO_INCREMENT`. Depois iremos torná-la nossa chave primária. Parece complicado, mas de fato não é. Na verdade, você fará tudo isso em um só comando:

*Desta vez estamos utilizando o comando CHANGE COLUMN, já que estamos alterando o nome e os tipos de dados da coluna anteriormente conhecida como numero.*

*Ainda estamos utilizando a mesma tabela, mas lembre-se, nós já demos um novo nome para ela.*

*id\_proj é o novo nome que queremos que nossa coluna tenha...*

*... e queremos que ela seja preenchida com valores inteiros tipo auto\_increment, valores NULL.*

```
ALTER TABLE lista_projetos
CHANGE COLUMN numero id_proj INT NOT NULL AUTO_INCREMENT,
ADD PRIMARY KEY ('id_proj');
```

*Aqui está a parte que diz ao nosso Sistema SQL para usar a coluna com o novo nome `id_proj` como a chave primária.*

 Aponte seu lápis

Rascunhe como a tabela ficará depois de você ter executado o comando acima.

Resposta na página 185.

## Altere duas colunas com apenas um comando SQL

Vamos alterar não uma, mas duas colunas com apenas um comando. Nós alteraremos os nomes das colunas chamadas de `descricaoProjeto` e `empreiteiroProjeto`, e ao mesmo tempo iremos alterar os tipos de dados. Tudo que precisamos fazer é incluir ambas as linhas `CHANGE COLUMN` em apenas um comando `ALTER TABLE` e colocar uma vírgula entre elas.

```
ALTER TABLE lista_projetos
CHANGE COLUMN descricaoProjeto descricao_proj VARCHAR(100),
CHANGE COLUMN empreiteiroProjeto empresa_nome VARCHAR(30);
```

*descricaoProjeto é o nome da coluna antiga que estamos alterando neste comando.*

*descricao\_proj é o novo nome da coluna.*

*Estamos aumentando o número de caracteres para que possamos ter descrições mais longas.*

*O outro nome antigo da coluna é empreiteiroProjeto, que também será alterado...*

*...para empresa\_nome e aqui está seu novo tipo de dados.*



**Veja Isto!**

**Se você alterar o tipo de dados para algo novo, poderá perder seus dados.**

**Se os tipos de dados que você está alterando não forem compatíveis com os tipos de dados antigos, seu comando não será considerado, e seu Sistema SQL irá dizer que tem um erro no seu comando.**

Mas a pior notícia é se elas são de tipos compatíveis, seus dados poderão ficar truncados.

Por exemplo: ir de `varchar(10)` para `char(1)` seus dados serão alterados de 'Bonzo' para apenas 'B'

A mesma regra se aplica aos tipos numéricos. Você pode alterar de um tipo para outro, mas seus dados serão convertidos para o novo tipo, e poderá perder parte de seus dados!



Quero alterar o tipo de dados de uma coluna, por exemplo, para armazenar tais caracteres, mas quero que o nome continue o mesmo. Posso repetir o nome da coluna, certo?

Desta forma:

```
ALTER TABLE minhaTabela  
CHANGE COLUMN minhaColuna minhaColuna  
TIPODEDADOS;
```

**Com certeza isso funcionaria, mas há um jeito mais simples.**

Você pode utilizar a palavra-chave MODIFY. Ela altera somente os tipos de dados de uma coluna e deixa o nome inalterado.

Por exemplo, suponha que precise de uma coluna mais longa para armazenar os dados da coluna descricao\_proj. Você quer que seja VARCHAR (120). Aqui está o que precisa fazer.

```
ALTER TABLE lista_projetos  
MODIFY COLUMN descricao_proj VARCHAR (120);
```

O nome da coluna que  
estamos modificando.

O novo tipo de dados.

*É claro que você certificou!!  
novo tipo de dados não vai trair  
seus dados antigos!*

## não existem Perguntas Idiotas

P: E se eu quiser alterar a ordem das colunas? Posso simplesmente utilizar: ALTER TABLE MODIFY COLUMN descricao\_proj AFTER nome\_empresa;

R: Você não pode, na verdade, alterar a ordem das colunas depois de criadas. O melhor que pode fazer é adicionar uma coluna na posição que desejar e eliminar a outra, mas irá perder todos os dados da coluna antiga.

P: Mas isso não vai ser um problema se as colunas estão armazenadas em uma ordem errada?

R: Não, porque, felizmente, nas suas consultas SELECT, você pode especificar a ordem em que suas colunas serão exibidas no resultado da consulta. Não importa a ordem que os dados sejam armazenados no seu disco, já que você pode utilizar:

```
SELECT coluna3, coluna1 FROM sua_tabela;
```

ou:

```
SELECT coluna1, coluna3 FROM sua_tabela;  
ou em qualquer ordem que desejar.
```



Exercícios 0



Hei! Estou no telefone  
com meu agente. Vá indo  
em frente e adicione  
aqueelas colunas restantes,  
você pode não é?

**lista\_projetos**

id_proj	descricao_proj	nome_empresa
1		
2		
3		

Precisamos de qualquer forma adicionar mais três colunas:  
**telefone, data de início e custo estimado.**

Escreva um comando ALTER TABLE simples abaixo,  
certificando-se de prestar atenção nos tipos de dados. Daí,  
complete a tabela abaixo.

.....  
.....  
.....  
.....

**lista\_projetos**



Solução dos Exercícios



Hei! Estou no telefone com meu agente. Vá indo em frente e adicione aquelas colunas restantes, você pode não é?

lista\_projetos

id_proj	descricao_proj	nome_empresa
1		
2		
3		

Precisamos de qualquer forma adicionar mais três colunas: **telefone, data de início e custo estimado.**

Escreva um comando ALTER TABLE simples abaixo, certificando-se de prestar atenção nos tipos de dados. Daí, complete a tabela abaixo.

Estamos adicionando novas colunas, então estamos utilizando o comando ADD.

ALTER TABLE tabela\_projeto

- ADD COLUMN empresa\_fone VARCHAR(10);
- ADD COLUMN data\_inicio DATE;
- ADD COLUMN custo\_est DECIMAL(7,2);

Um VARCHAR de 10 campos permite que você adicione o código da área.

Lembre-se dos nossos campos DEC? Nós definimos esta coluna para que o número inteiro tenha caracteres e os dígitos 2 casas decimais.

lista\_projetos


## Rápido! Largue esta coluna!

Pare tudo!

Acabamos de descobrir que nossos projetos foram colocados em espera. Consequentemente, podemos eliminar a coluna `data_inicio`. Não há razão em ter uma coluna desnecessária encostada e ocupando espaço no banco de dados.

É uma boa prática de programação ter apenas as colunas que precisa na tabela. Se não estiver utilizando uma coluna, elimine-a. Com ALTER, você pode facilmente adicioná-la de volta, caso precise dela no futuro.

Quanto mais colunas tenha, mais o seu Sistema SQL tem que trabalhar, e mais espaço seu bancos de dados vão ocupar. Enquanto pode não perceber isto com uma pequena tabela, quando ela crescer, você verá resultados mais lentos, e o processador de seu computador terá que trabalhar tão duro quanto.



## Aponte seu lápis

Pra falar a verdade, vai em frente e escreva seu comando SQL para eliminar a coluna data\_inicio. Ainda não lhe mostramos a sintaxe para fazê-la ainda, mas faça uma tentativa.

## Aponte seu lápis

### Solução

Pra falar a verdade, vai em frente e escreva seu comando SQL para eliminar a coluna data\_inicio. Ainda não lhe mostramos a sintaxe para fazê-la ainda, mas faça uma tentativa.

Aqui está nossa lista\_de\_projetos.

**ALTER TABLE lista\_projetos**

**DROP COLUMN data\_inicio;**

Se quer eliminar a coluna  
data\_inicio, você pode  
utilizar o comando DROP.  
Isso foi fácil!

A coluna a ser  
removida da tabela



Depois de ter eliminado uma coluna, tudo o que estava armazenado será removido também!

### Veja Isto!

Use o comando `DROP COLUMN` cautelosamente. Primeiro deve utilizar o comando `SELECT` a partir daquela coluna que pretende eliminar para ter certeza absoluta que é isso que quer! É melhor ter dados extras na sua tabela do que depois faltar dados vitais na sua tabela.

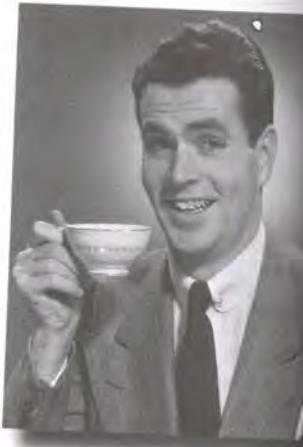


# Tabela minha Turbine

É hora de transformar sua tabela de carro usado em uma tabela chamariz e levá-la a outro nível de turbinagem de tabela que você nunca antes havia experimentado.

É simples. Pegue esta tabela patética de "antes" com dados de carros usados e use o ALTER naquela fantástica, brilhosa tabela de "depois". Parte da dificuldade é não perturbar quaisquer dos dados na tabela, mas trabalhar ao redor deles. Você está apto ao desafio?

Pontos de bônus se conseguir fazer tudo isso com apenas um comando ALTER.



## Antes

usado

cor	ano	fabricante	mod	valordecusto
prata	1998	Porsche	Boxter	17992.540
NULL	2000	Jaguar	XJ	15995
vermelho	2002	Cadillac	Escalade	40215.9

## Depois

tabela\_carro

carro_id	chassi	fabricante	modelo	cor	ano	valor
1	RNKLK66N33G213481	Porsche	Boxter	prata	1998	17992.54
2	SAEDA44B175B04113	Jaguar	XJ	NULL	2000	15995.00
3	3GYEK63NT2G280668	Cadillac	Escalade	vermelho	2002	40215.90



Exercícios

# Turbine minha Tabela

É hora de transformar sua tabela de carro usado em uma tabela chamariz e levá-la a outro nível de turbinagem de tabela que você nunca antes havia experimentado.

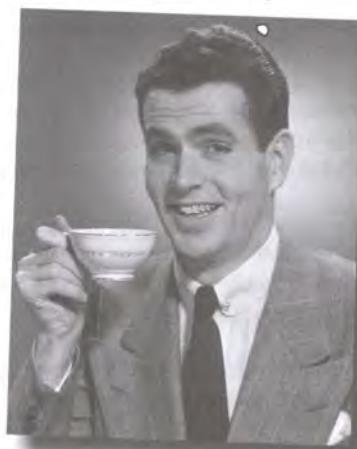
É simples. Pegue esta tabela patética de "antes" com dados de carros usados e use o ALTER naquela fantástica, brilhosa tabela de "depois". Parte da dificuldade é não perturbar quaisquer dos dados na tabela, mas trabalhar ao redor deles. Você está apto ao desafio?

Pontos de bônus se conseguir fazer tudo isso com apenas um comando ALTER.

## Antes

usado

cor	ano	fabricante	mod	valordecusto
prata	1998	Porsche	Boxter	17992.540
NULL	2000	Jaguar	XJ	15995
vermelho	2002	Cadillac	Escalade	40215.9



## Depois

tabela\_carro

carro_id	chassi	fabricante	modelo	cor	ano	valor
1	RNKLK66N33G213481	Porsche	Boxter	prata	1998	17992.54
2	SAEDA44B175B04113	Jaguar	XJ	NULL	2000	15995.00
3	3GYEK63NT2G280668	Cadillac	Escalade	vermelho	2002	40215.90

ALTER TABLE usado

RENAME TO tabela\_carro;

ALTER TABLE tabela\_carro

ADD COLUMN carro\_id INT NOT NULL AUTO\_INCREMENT FIRST,

ADD PRIMARY KEY (carro\_id),

ALTER TABLE tabela\_carro

ADD COLUMN CHAST VARCHAR(16) SECOND,

CHANGE COLUMN modelo VARCHAR(20),

MODIFY COLUMN cor AFTER modelo,

MODIFY COLUMN ano SIXTH,

CHANGE COLUMN valordecusto valor DECIMAL(7,2);

Você poderia ter utilizado o DESCRIBE primeiro, para que pudesse ver os tipos de dados de cada coluna para ter certeza que não estaria truncando os dados.

Você precisa renomear a coluna chamada mod para modelo antes de mover as colunas cor e ano para depois dela.

Você já deve dar para a sua coluna renomeada para modelo o tipo de dados.

Você poderia também ter colocado ano AFTER modelo ou ano BEFORE valor.

não existem  
Perguntas Idiotas

**P:** Anteriormente você disse que eu não conseguia reordenar minhas colunas com MODIFY. Mas as ferramentas de meu Sistema SQL me permitem reordená-las. Como ele está fazendo isso?

**R:** Seu Sistema está fazendo uma porção de comandos por detrás das cortinas. Ele está copiando os valores da coluna que você gostaria de mover, salvando em uma tabela temporária, eliminando a coluna com o mesmo nome daquela de origem, alterando sua tabela e criando uma nova coluna com o mesmo nome, tal como a antiga no lugar onde você gostaria que ela estivesse, copiando todos os valores da tabela temporária para sua nova coluna e deletando a tabela temporária.

Geralmente é melhor apenas manter a posição da coluna do jeito que está no caso delas já conterem dados inseridos e você não utilizar o Sistema para ele fazer todos os passos por você. SELECT (selecionar) suas colunas na ordem que desejar.

**P:** O único momento em que é fácil alterar a ordem da coluna é quando estou criando uma nova coluna?

**R:** Correto. A melhor opção é pensar sobre a ordem no momento que você iniciar o projeto da sua tabela.

**P:** E se eu criei uma chave primária acidentalmente, e então mudar de idéia e pretender utilizar uma outra coluna? Há algum jeito de remover a designação da chave primária sem alterar os dados ali inseridos?

**R:** Há sim, e é bem simples:

```
ALTER TABLE sua_tabela DROP PRIMARY KEY;
```

**P:** E quanto ao AUTO\_INCREMENT?

**R:** Você pode adicioná-lo a uma coluna que não tem este atributo, da seguinte forma:

```
ALTER TABLE sua_tabela CHANGE sua_chave
sua_chave INT (11) NOT NULL AUTO_INCREMENT;
```

E você pode remover assim:

```
ALTER TABLE sua_tabela CHANGE sua_chave
sua_chave INT (11) NOT NULL;
```

É importante manter em mente que você pode ter somente um campo AUTO\_INCREMENT por tabela e ele deve ser do tipo INTEGER e não pode conter dados do tipo NULL.

### PONTOS DE BALA



- Use CHANGE quando quiser alterar o nome e os tipos de dados de uma coluna.
- Use MODIFY quando desejar alterar apenas o tipo de dados.
- DROP COLUMN tem apenas esta função: eliminar a coluna selecionada da tabela.
- Use RENAME para alterar o nome da sua tabela.

- Você pode alterar a ordem das suas colunas usando FIRST, LAST, BEFORE nome\_coluna, AFTER nome\_coluna, SECOND, THIRD, FOURTH, etc.
- Com alguns sistemas SQL você pode alterar a ordem das colunas, apenas quando as insere na tabela.



### ALTER TABLE pode ajudá-lo a alterar seu projeto de tabela

Ao utilizar ALTER TABLE junto com SELECT e UPDATE, podemos pegar colunas de dados estranhas e não-atômicas e refiná-las a colunas atômicas precisas. É tudo uma questão de combinar comandos SQL que você já aprendeu, nas formas corretas.

Vamos dar uma olhada no comando CREATE TABLE para a tabela meus\_contatos de Greg.

Adicionamos  
estas duas  
linhas para  
e designar n  
chave primár

Um olha

As vezes, Gre

Estes da  
formato  
da cidad  
depois a  
do esta  
dos dad  
cidades

```

CREATE TABLE meus_contatos
(
    id_contato INT NOT NULL AUTO_INCREMENT,
    sobrenome VARCHAR(30) default NULL,
    primeiro_nome VARCHAR(20) default NULL,
    email VARCHAR(50) default NULL,
    sexo CHAR(1) default NULL,
    aniversario DATE default NULL,
    profissao VARCHAR(50) default NULL,
    local VARCHAR(50) default NULL, ←
    estado_civil VARCHAR(20) default NULL ←
    interesses VARCHAR(100) default NULL, ←
    procura VARCHAR(100) default NULL, ←
    PRIMARY KEY (id_contato)
)

```

*Adicionamos  
estas duas  
linhas para criar  
uma chave primária.*

*Estas quatro colunas  
não são muito atômicas  
e poderiam usar alguns  
ajustes feitos pelo  
ALTER TABLE.*

## Um olhar de perto para a coluna não-atômica "local"

Veja, Greg quer saber apenas o estado ou cidade de alguém, então a coluna local é uma candidata a ser dividida em duas colunas. Vamos ver como os dados nas colunas estão:

File	Edit	Window	Help	Location	Location	Location
--> SELECT local FROM meus_contatos;						
+	-	-	-	+	-	-
	local					
+	-	-	-	+	-	-
	Seattle, WA					
	Natchez, MS					
	Las Vegas, NV					
	Palo Alto, CA					
	NYC, NY					
	Phoenix, AZ					

Um pedacinho dos dados da coluna local da tabela meus\_contatos.

Nome da cidade.

Seattle, WA

Natchez, MS

Las Vegas, NV

Palo Alto, CA

NYC, NY

Abreviação de duas letras para o estado.

uma vírgula.

Estes dados estão consistentemente formatados. Primeiro vem o nome da cidade, seguido de uma vírgula e depois a abreviação de duas letras do estado. Devido à consistência dos dados, poderemos separar as cidades dos estados.



### Musculação cerebral

Por que nós queremos separar a cidade do estado?  
O que acha que faremos depois?

## Procure por padrões

Cada coluna “local” na tabela `meus_contatos` segue os mesmos padrões: Nome da cidade, seguida de uma vírgula e então a abreviação de duas letras do estado, ou Unidade Federativa (UF). O fato de ela ser consistente e seguir um padrão nos ajudará a dividi-la para que fique mais atômica.

**Nome da Cidade, XX**

Esta vírgula que sempre vem antes do estado pode ser uma mão na roda...

Podemos pegar tudo aquilo que está antes da vírgula e colocar em uma coluna contendo nomes das cidades.

**Nome da Cidade**

Precisamos de uma função que nos permita pegar tudo antes da vírgula...



Aponte seu lápis —

Estes dois últimos caracteres sempre contêm a abreviação para o estado. Se tivéssemos uma coluna para estado na nossa tabela, estes seriam os dados que iríamos utilizar.

E podemos pegar os dois últimos caracteres da coluna e colocá-los em uma nova coluna chamada estado.

**XX**

...E nós precisamos de uma função que irá pegar os dois últimos caracteres.

Escreva um comando `ALTER TABLE` que adiciona as colunas cidade e estado para `meus_contatos`.

```
ALTER TABLE meus_contatos
ADD COLUMN cidade VARCHAR(50),
ADD COLUMN estado CHAR(2);
```

## Algumas funções de texto muito úteis

Localizamos dois padrões muito importantes. Agora precisamos agarrar a abreviação do estado e adicioná-la em uma nova coluna `estado`. Precisamos também de tudo que esteja antes da vírgula para uma coluna `cidade`. Depois de criarmos nossas duas colunas, agora como extrair os valores que precisávamos:

### SELECT (selecionar) nossos dois últimos caracteres

Use `RIGHT()` e `LEFT()` para selecionar uma quantidade específica de caracteres de uma coluna.

**SELECT RIGHT(local, 2) FROM meus\_contatos;**

Comece do lado direito da coluna (você pode utilizar `LEFT` para esquerda, da mesma maneira).

Esta é a coluna a ser utilizada.

Esta é a quantidade de caracteres a serem selecionados do lado direito da coluna.

Valores de texto:  
valores armazenados em colunas CHAR e VARCHAR são conhecidos como de texto, ou strings.

## Selecionar tudo à frente da vírgula

use `SUBSTRING_INDEX()` para pegar parte da coluna, ou seja. Este comando irá encontrar tudo que estiver **à frente** de uma linha de texto específica. Então, podemos colocar nossa vírgula entre aspas, e `SUBSTRING_INDEX()` irá selecionar tudo que estiver à frente dela.

```
SELECT SUBSTRING_INDEX(local, ',', 1) FROM meus_contatos;
```

Este comando pega parte da coluna, ou seja, procura pela linha de texto entre aspas (neste caso é a vírgula) e consegue pegar tudo que esteja à sua frente.

Novamente, o nome da coluna.

Aqui está a vírgula pela qual o programa está procurando.

Funções string permitem selecionar parte do texto de uma coluna.

1) `FROM meus_contatos;`

Esta é a parte confusa. Aqui diz / porque está procurando pela primeira vírgula.

## Tente em casa



Exercícios

SQL possui uma infinidade de funções que permitem que você manipule valores de texto nas suas tabelas. Linhas de texto são armazenadas nas colunas de texto, tipicamente dados do tipo VARCHAR ou CHAR.

Aqui está uma lista das funções mais comuns e úteis. Tente cada uma delas por conta própria digitando seus comandos SELECT.

`SUBSTRING(sua_linha_de_texto, posicao_de_inicio, tamanho)` fornece parte da sua linha de texto, iniciando pela letra na posição de inicio. Tamanho é o quanto do texto você quer selecionado.

```
SELECT SUBSTRING ('San Antonio, TX', 5, 3);
```

`UPPER (sua_linha_de_texto)` e `LOWER (sua_linha_de_texto)` muda tudo na sua string para caixa alta ou baixa, respectivamente.

```
SELECT UPPER ('uSa');
```

```
SELECT LOWER ('spaGHetti');
```

`REVERSE (sua_linha_de_texto)` faz justamente o que diz, ela reverte a ordem de letras na sua linha de texto.

```
SELECT REVERSE ('SpaGHEtti');
```

`LTRIM (sua_linha_de_texto)` e `RTRIM (sua_linha_de_texto)` retornam sua string espaços extras removidos de antes (à esquerda) ou depois (à direita) da string.

```
SELECT LTRIM(' comida_cachorro');
```

```
SELECT RTRIM(' comida_gata');
```

`LENGTH (sua_linha_de_texto)` retornam uma conta de quantos caracteres existem na sua linha de texto.

```
SELECT LENGTH ('San Antonio, TX');
```

**IMPORTANTE: funções String não alteram os dados inseridos na sua tabela, elas simplesmente retornam as strings alteradas como um resultado de sua consulta.**

## QUAL É O MEU PROPÓSITO?

Estamos tentando pegar as informações contidas na nossa coluna local e transferi-las para duas novas colunas, cidade e estado.

Aqui estão os passos que precisaremos seguir para fazer isto. Combine cada passo necessário com a palavra-chave ou as palavras-chave que precisamos para concluir aquela etapa em particular.

**SUBSTRING\_INDEX()**

**SELECT**

- 1. Dê uma olhada nos dados de uma coluna em particular para tentar encontrar um padrão.**

**ADD COLUMN**

**LEFT**

**RIGHT**

- 2. Adicione novas colunas vazias para sua tabela.**

**ADJUST**

**DELETE**

- 3. Pegue parte dos dados de uma coluna de texto.**

**ALTER TABLE**

**UPDATE**

**INSERT**

— Resposta na próxima página



Sabemos como usar todos os pedaços de forma correta, mas ainda não sabemos como juntá-los eficientemente. Talvez pudéssemos tentar utilizar aquelas funções strings com um comando UPDATE...

**Com que o que nós sabemos até agora, teríamos que utilizar o comando UPDATE, em um registro de cada vez, com o SELECT para selecionar os dados corretos.**

Mas com SQL, podemos combinar comandos. Veja na próxima página como colocar os valores nas nossas novas colunas.

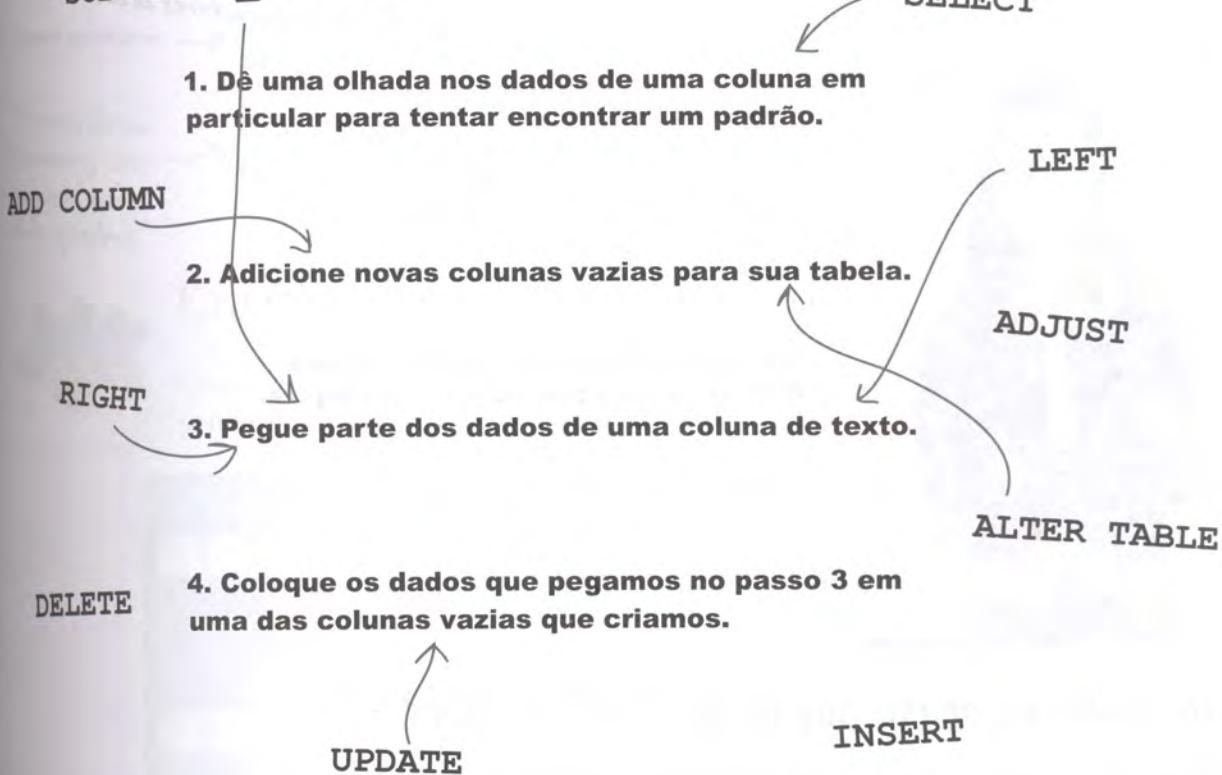
**Util**  
Lembre-  
conter o  
**em um**

## QUAL É O MEU PROPÓSITO?

Estamos tentando pegar as informações contidas na nossa coluna local e transferi-las para duas novas colunas, cidade e estado.

Aqui estão os passos que precisaremos seguir para fazer isto. Combine cada passo necessário com a palavra-chave ou as palavras-chave que precisamos para concluir aquela etapa em particular.

### SUBSTRING\_INDEX()



### Utilize uma coluna atual para preencher uma nova coluna

Lembre-se da sintaxe do comando `UPDATE`? Podemos utilizá-lo para definir cada registro na nossa tabela para ter o mesmo valor. O comando abaixo mostra a sintaxe utilizada para **alterar o valor de cada registro em uma coluna**. Em vez de um novo valor, você pode inserir um outro valor ou outro nome de coluna.

```
UPDATE nome_tabela
SET nome_coluna = novovalor;
```

*Cada registro em nossa tabela está definido, um de cada vez, para este valor.*

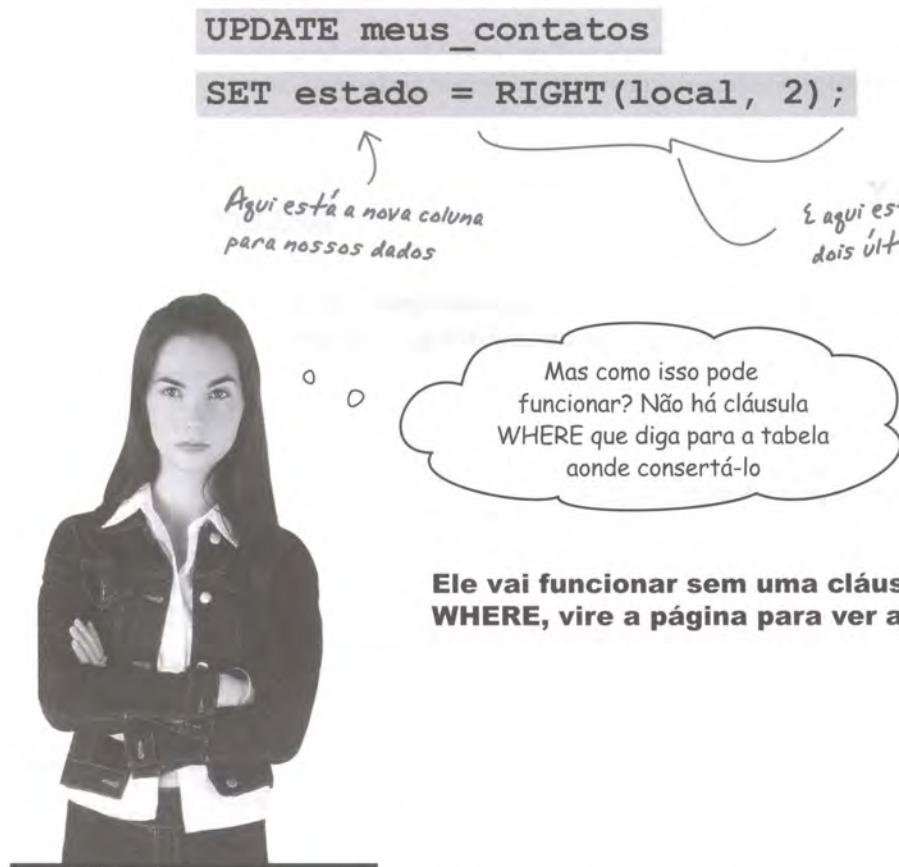
Para inserir dados para nossas colunas `cidade` e `estado`, podemos utilizar a função de string `RIGHT()` dentro do comando `UPDATE`. A função `String` pega os dois últimos caracteres da antiga coluna local e os coloca em uma nova coluna.

Vamos an  
primeira v  
primeira c  
Então, ele  
vez, encor  
que todos  
nenhum r

Prim

Seg

Ten  
var  
por  
trê



**Ele vai funcionar sem uma cláusula WHERE, vire a página para ver agora.**

## Como funciona nosso combo UPDATE e SET?

Seu Sistema SQL interpreta o comando para cada linha da tabela e por vez; depois ele volta ao início até que todas as abreviações de estados estejam divididas para a nova coluna `estado`.

meus_contatos			
<code>id_contato</code>	<code>local</code>	<code>cidade</code>	<code>estado</code>
1	Chester, NJ		
2	Katy, TX		
3	San Mateo, CA		

Agui está uma versão simplificada de nossa

Horizontais  
2. \_\_\_ (sua esquerda)  
4. Nossa ta  
cláusula CO  
6. \_\_\_ (su  
8. ALTER T  
9. Você po  
10. SUBST  
começando  
que seja re  
11. Use \_\_\_

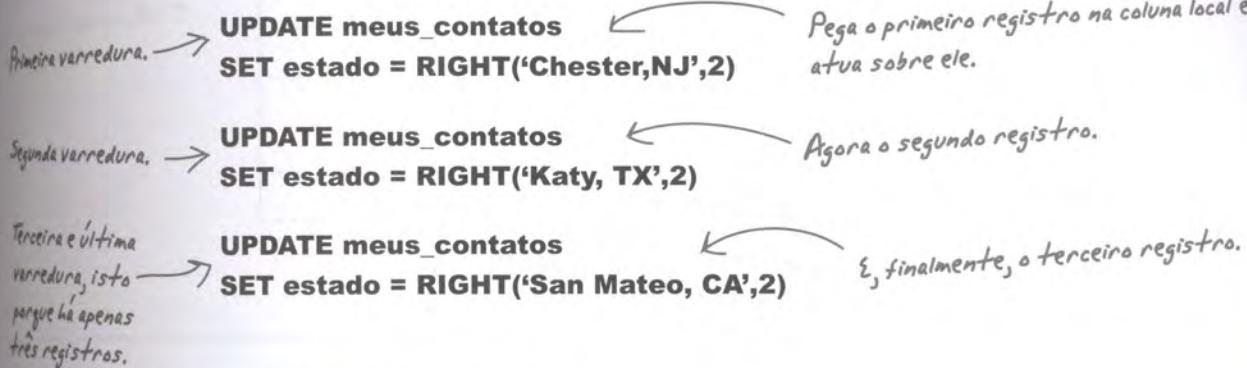
```
UPDATE meus_contatos
SET estado = RIGHT (local, 2);
```

Agui está nosso comando SQL

analisar como age a instrução anterior nesta tabela de exemplo. Na primeira vez em que a tabela é analisada, ele leva a coluna local para a mesma coluna e trabalha sobre ela.

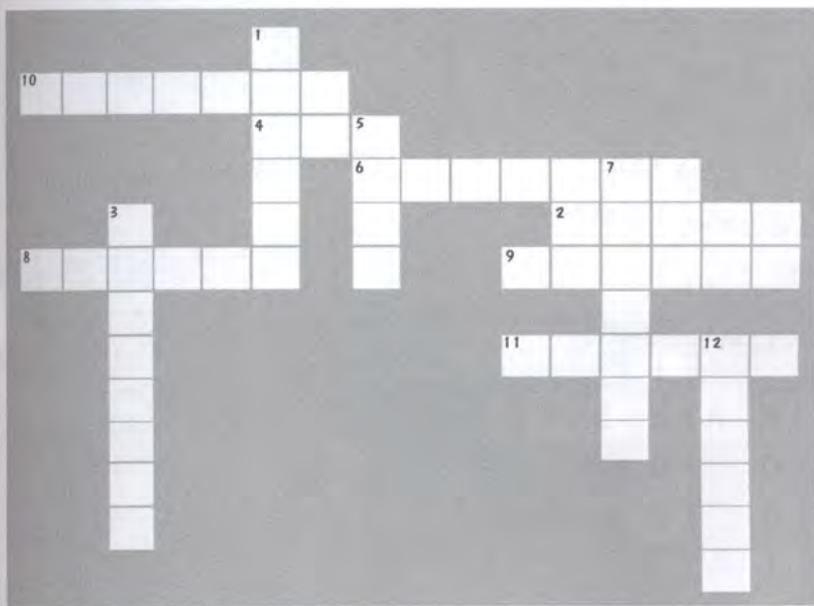
Depois, ele começa a correr sobre a tabela inteira novamente por uma segunda vez, encontrando o local na segunda linha, atua nela, e assim sucessivamente, até que todos os registros referentes ao estado sejam divididos e não haja mais nenhum registro que se ajuste ao comando.

Nós podemos utilizar funções strings em conjunto com SELECT, UPDATE e DELETE.



## Cruzadas-ALTER

Como um jogo de palavras cruzadas te ajuda a aprender SQL? Bem, ele o faz pensar nos comandos e palavras-chave deste capítulo de uma forma diferente.



**versão**  
**nossa tabela.**

- 1. `LTRIM(sua_string)` exibe sua string com espaços demais removidos antes (ou à esquerda) de uma string.
- 2. Esta tabela pode ganhar novas colunas com o comando ALTER e \_\_\_\_\_ uma `ADD COLUMN`.
- 3. `REVERSE(sua_string)` faz justamente isso, ela reverte a ordem das letras na sua string.
- 4. `ALTER TABLE projetojs ____ TO lista_projetos;`
- 5. Você pode utilizar funções \_\_\_\_\_ combinadas com SELECT, UPDATE e DELETE.
- 6. `SUBSTRING(sua_string, posicao_inicial, tamanho)` fornece parte de sua\_string, ignorando a letra indicada por posicao\_inicial. \_\_\_\_\_ é o quanto da string você quer que seja retornado.
- 7. Use \_\_\_\_\_ para alterar o nome da sua tabela.

### Verticais

1. Use esta palavra-chave para alterar o tipo de dados armazenados em uma coluna.
3. Você pode ter apenas um campo `AUTO_INCREMENT` por tabela, ele tem que ser do tipo de dados \_\_\_\_\_.
5. Quando não mais precisar de uma coluna, utiliza \_\_\_\_\_ `COLUMN` com `ALTER`.
7. Valores armazenados em colunas CHAR ou VARCHAR são conhecidos assim.
12. Utilize esta cláusula com `ALTER` quando desejar apenas alterar os tipos de dados.



## Sua caixa de ferramenta SQL

Dê uma mãozinha a você mesmo.

Você já é mestre no Capítulo 5, e agora adicionou ALTER para sua caixa de ferramentas. Para uma lista completa de dicas neste livro, veja o Apêndice iii.

### ALTER TABLE

Permite alterar o nome de sua tabela e toda sua estrutura e ainda preserva os dados inseridos nela.

### ALTER com CHANGE

Permite alterar o nome e o tipo de dados de uma coluna já existente.

### ALTER com MODIFY

Permite alterar apenas o tipo de dados de uma coluna existente.

### ALTER com ADD

Permite adicionar uma coluna à sua tabela na ordem que você escolher.

### ALTER com DROP

Permite eliminar uma coluna da sua tabela.

### Funções String

Permite modificar cópias dos conteúdos de uma coluna quando elas são exibidas em uma consulta. Os valores originais são mantidos intocáveis.

 Aponte seu lápis –  
da página 169.

Rascunhe como a tabela ficará depois de você ter executado o comando acima.

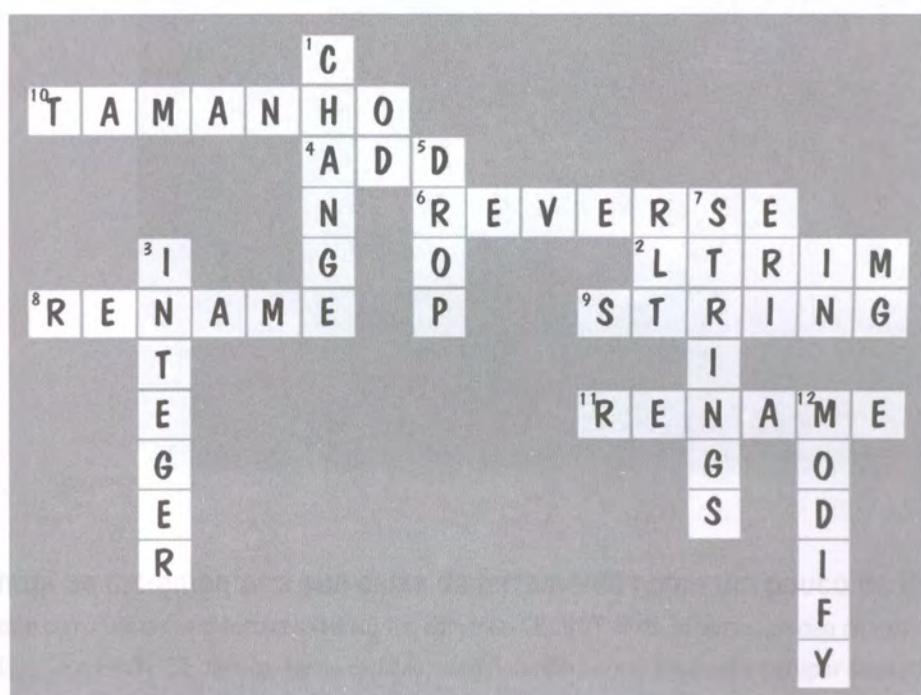
O terceiro número se  
tornou id\_proj e aquela  
coluna contém valores  
tipo chave primária AUTO  
INCREMENT.

## lista\_projetos

<u>id_proj</u>	<u>descricao_proj</u>	<u>nome_empresa</u>
1	pintar exterior da casa	Murphy
2	remodelar a cozinha	Valdez
3	instalar piso de madeira na cozinha	Keller
4	trocar o telhado	Jackson



## Solução das Cruzadas-ALTER





## 6 SELECT avançado

### Ver os seus dados com novos olhos



E então fui capaz de ver  
apenas os aviões inimigos  
usando o comando CASE!  
Bang!



**É hora de incrementar a sua caixa de ferramentas com um pouco de finesse.** Você já sabe como utilizar as cláusulas WHERE e o comando SELECT. Mas, às vezes, precisa de mais exatidão que o SELECT e o PROVIDE oferece. Neste capítulo, você aprenderá como **ordenar e agrupar** seus dados, bem como realizar **operações matemáticas** nos seus resultados.

# Dataville Video está se reorganizando

O proprietário da Dataville Video tem uma loja desorganizada. Em seu sistema atual, os filmes podem acabar ficando em prateleiras diferentes dependendo de qual funcionário for guardá-lo. Ele já comprou novas prateleiras e acredita ser uma ótima hora para, finalmente, organizar cada gênero de filme.



No sistema atual, valores Verdadeiro/Falso são utilizados para as categorias de filmes. Isto faz com que seja difícil classificar um filme. Por exemplo, se um filme tem V para comédia e um V para Ficção Científica, onde ele deveria ser armazenado?

Para: Equipe do Dataville Video

De: Chefe

Assunto: Novas prateleiras significam novas categorias!

Oi galera,

As novas prateleiras estão aí, então quero organizar os filmes. Nós podemos utilizar as seguintes categorias:

Ação e Aventura

Drama

Comédia

Família

Terror

Ficção Científica e Fantasia

Diversos

Vou deixar por conta de vocês imaginar como fazer nossa tabela atual funcionar com estas novas categorias.

Vamos almoçar,

O chefe.

↓  
Aqui está a data em que a loja comprou a cópia.

"V" e "F" são abreviações para Verdadeiro e Falso.

tabela\_filme

id_filme	titulo	censura	drama	comedia	acao	horror	ficcao	infantil	desenho	data
1	Monsters, Inc.	0	F	V	F	F	F	V	V	1
2	The Godfather	18	F	F	V	V	F	F	F	2
3	Gone with the Wind	0	V	F	F	F	F	F	F	20
4	American Pie	18	F	V	F	F	F	F	F	14
5	Nightmare on Elm Street	18	F	F	V	V	F	F	F	16
6	Casablanca	13	V	F	F	F	F	F	F	3

Todas estas colunas existem para que possamos responder às perguntas dos clientes sobre o conteúdo de um filme em particular.

## Problemas com nossa tabela atual

Está uma revisão dos problemas que Dataville Video tem com sua tabela atual.

**Quando os filmes são devolvidos, não sabemos em qual prateleira ele deve ficar.**

Se temos valores V para um número de colunas na tabela, não há um jeito claro de saber onde o filme deve ser guardado. Os filmes devem sempre estar associados **a uma só categoria**.

**As pessoas não estão esclarecidas sobre o que o filme se trata.**

Nossos clientes se confundem quando vêem um filme com capa de terror em uma seção de comédia. Atualmente nenhum dos valores V/F tem prioridade sobre qualquer outra categoria quando os filmes são guardados.

**Inserir os dados Verdadeiro ou Falso demora muito e erros acontecem freqüentemente.**

Toda vez que um novo filme chega, ele deve ser cadastrado com todas as colunas V/F. E quanto mais filmes são inseridos, mais erros são agrupados. Às vezes uma coluna que deveria ter um valor V, acidentalmente, é inserida com um valor F, e vice-versa. Uma coluna para categoria faria com que checássemos duas vezes nossas colunas V/F e, eventualmente, poderíamos eliminar os valores V/F todos de uma vez.

**O que precisamos aqui é uma coluna categoria para acelerar o armazenamento, ajudar os clientes a descobrir que tipo de filme eles estão locando, e limitar os erros de sua tabela.**



**PODER DO  
CÉREBRO**

Como você organizaria as colunas atuais em novas categorias? Há filmes que se encaixam em mais de uma das novas categorias criadas?

## Combinando dados existentes

Você sabe como usar o ALTER em sua tabela para adicionar uma nova coluna categoria, mas adicionar as atuais categorias é um pouquinho mais complicado. Com sorte, os dados que já estão inseridos na tabela podem nos ajudar a imaginar a categoria para cada filme, sem que tenhamos que olhar registro por registro.

Vamos rescrever os relacionamentos em sentenças simples:

- Se esta coluna é 'V': **drama** definimos a categoria como 'drama'
- Se esta coluna é 'V': **comedia** definimos a categoria como 'comédia'
- Se esta coluna é 'V': **acao** definimos a categoria como 'ação'
- Se esta coluna é 'V': **horror** definimos a categoria como 'terror'
- Se esta coluna é 'V': **ficcao** definimos a categoria como 'ficção'
- Se esta coluna é 'V': **infantil** definimos a categoria como 'família'
- Se esta coluna é 'V': **desenho** e esta coluna é '0': **censura** definimos a categoria como 'família'
- Se esta coluna é 'V': **desenho** e esta coluna NÃO é '0': **censura** definimos a categoria como 'diversos'

*Nem todos os desenhos são para crianças. A coluna censura ajuda a determinar se o filme está na categoria família ou não, dependendo se o valor é Verdadeiro ou Falso. Se a censura for 0, podemos chamar o filme de familiar, caso contrário é da categoria diversos.*

## Povoando sua nova coluna

Agora podemos traduzir estas sentenças em comandos UPDATE do SQL:

```
UPDATE tabela_filme SET categoria = 'drama' where drama = 'V';
UPDATE tabela_filme SET categoria = 'comédia' where comedia = 'V';
UPDATE tabela_filme SET categoria = 'ação' where acao = 'V';
UPDATE tabela_filme SET categoria = 'terror' where horror = 'V';
UPDATE tabela_filme SET categoria = 'fíção' where ficcao = 'V';
UPDATE tabela_filme SET categoria = 'família' where infantil = 'V';
UPDATE tabela_filme SET categoria = 'família' where desenho = 'V' AND censura = 0;
UPDATE tabela_filme SET categoria = 'diversos' where desenho = 'V' AND censura > 0;
```

*Censura não é igual a 0*

 Aponte seu lápis

Preencha os valores para a coluna categoria em relação aos filmes abaixo

tabela\_filme

titulo	censura	drama	comedia	acao	horror	ficcao	infantil	desenho	categoria
Big Adventure	0	F	F	F	F	F	V	F	
Greg: The Untold Story	13	F	F	V	F	F	F	F	
Mad Clowns	18	F	F	F	V	F	F	F	
Paraskavedekatriaphobia	18	V	V	V	F	V	F	F	
Rat named Darcy, A	0	F	F	F	F	F	V	F	
End of the Line	18	V	F	F	V	V	F	V	
Shiny Things, The	13	V	F	F	F	F	F	F	
Take it Back	18	F	V	F	F	F	F	F	
Shark Bait	0	F	F	F	F	F	V	F	
Angry Pirate	13	F	V	F	F	F	F	V	
Potentially Habitable Planet	13	F	V	F	F	V	F	F	

A ordem que avaliamos cada coluna V/F importa? .....

 Aponte seu lápis

Solução

Preencha os valores para a coluna categoria em relação aos filmes abaixo

tabela\_filme

titulo	censura	drama	comedia	acao	horror	ficcao	infantil	desenho	categoria
Big Adventure	0	F	F	F	F	F	V	F	família
Greg: The Untold Story	13	F	F	V	F	F	F	F	ação
Mad Clowns	18	F	F	F	V	F	F	F	terror
Paraskavedekatriaphobia	18	V	V	V	F	V	F	F	?
Rat named Darcy, A	0	F	F	F	F	F	V	F	família
End of the Line	18	V	F	F	V	V	F	V	diversos
Shiny Things, The	13	V	F	F	F	F	F	F	drama
Take it Back	18	F	V	F	F	F	F	F	comédia
Shark Bait	0	F	F	F	F	F	V	F	?
Angry Pirate	13	F	V	F	F	F	F	V	diversos
Potentially Habitable Planet	13	F	V	F	F	V	F	F	?

O ponto de interrogação quer dizer que uma coluna foi alterada por mais de um UPDATE.   
Este valor será alterado dependendo da ordem que o UPDATE tem para ser executado.

A ordem que avaliamos cada coluna V/F importa? Sim ela importa.....

## A ordem importa de fato

Por exemplo, se para nós o filme Paraskavedekatriaphobia fosse classificado como ficção, ainda que ele seja mais do gênero comédia, não sabemos se ele deve ser considerado comédia, ação, drama, desenho ou ficção. Tendo em vista que ainda não seja claro de onde eles vieram, o melhor lugar para ele seria a coluna categoria diversos.



Isto parece ótimo para uma tabela pequena, mas e se você tivesse centenas de colunas? Seria possível combinar todos os comandos UPDATE em um só grande comando?

Ordem importa

Dois comandos UPDATE podem alterar o mesmo valor de coluna.

UPDA

Vamos ver

UPI

SET

CAS

VI

VI

VI

VI

VI

VI

E

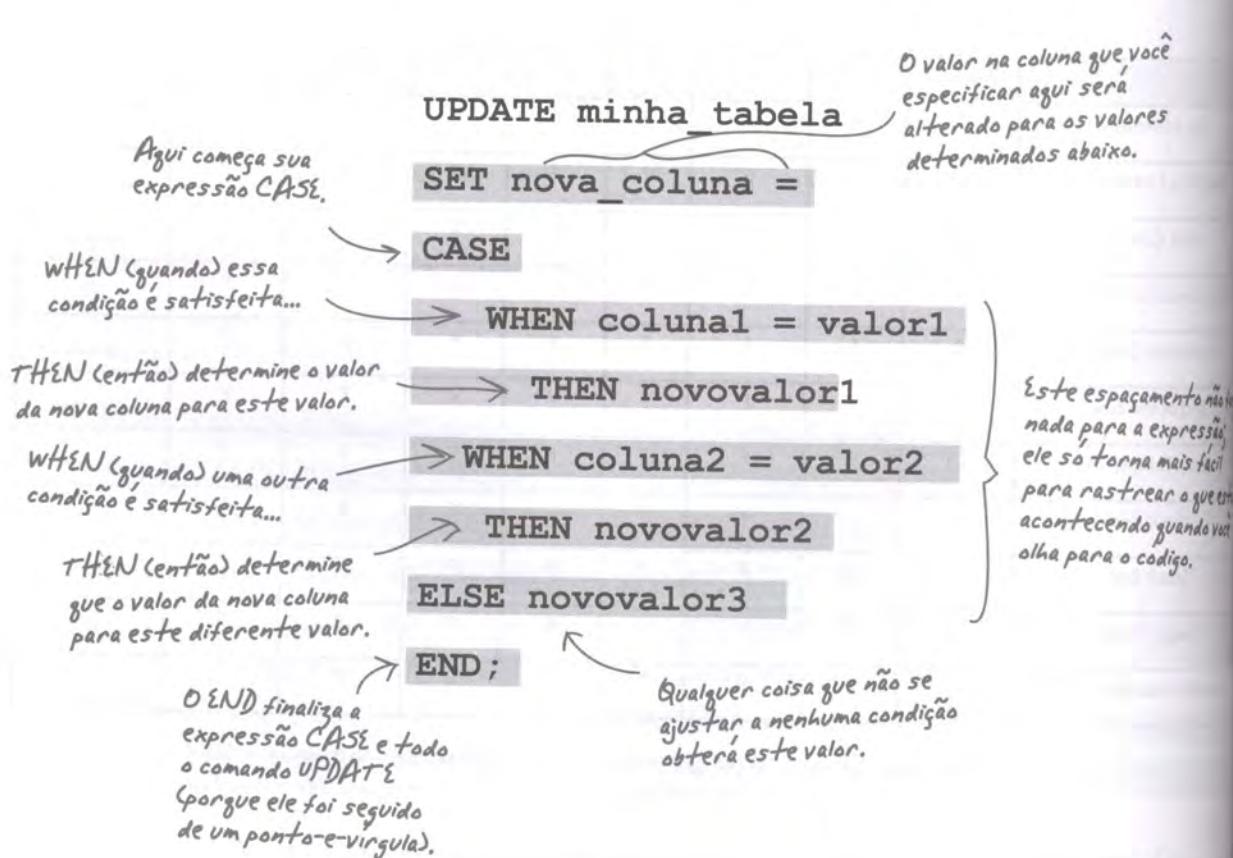
END

**Bem, você poderia escrever um só grande comando UPDATE, mas há um jeito ainda melhor.**

A expressão CASE combina todos os comandos UPDATE ao checar um valor de coluna existente contra uma condição. Se ele se encaixa na condição, a nova coluna é preenchida com o valor especificado.

Ele permite até que você diga ao seu Sistema SQL **o que fazer se alguns valores não satisfizerem a condição**.

Aqui está sua sintaxe básica:



## UPDATE com uma expressão CASE

Ver a expressão CASE em ação na nossa tabela\_filme.

```
UPDATE tabela_filme
SET categoria =
CASE
    WHEN drama = 'V' THEN 'drama'
    WHEN comedia = 'V' THEN 'comédia'
    WHEN acao = 'V' THEN 'ação'
    WHEN horror = 'V' THEN 'terror'
    WHEN ficcao = 'V' THEN 'ficção'
    WHEN infantil = 'V' THEN 'família'
    WHEN desenho = 'V' THEN 'família'
    ELSE 'diversos'
END;
```

Tudo o que não satisfizer as condições nas linhas acima é atribuído como da categoria diversos.

Isto é o mesmo que dizer UPDATE tabela\_filme SET categoria = drama WHERE drama = 'V' - mas com muito menos digitação!

Os valores que eram desconhecidos quando utilizamos UPDATE neles mesmos para povoar a nova coluna agora têm valores de categoria.

Mas perceba como agora também temos novos valores para Angry Pirate e End of the Line

tabela\_filme

título	censura	drama	comedia	acao	horror	ficcao	infantil	desenho	categoria
Big Adventure	0	F	F	F	F	F	V	F	Família
Greg: The Untold Story	13	F	F	T	F	F	F	F	Ação
Mad Clowns	18	F	F	F	V	F	F	F	terror
Paraskavedekatriaphobia	18	V	V	V	F	V	F	F	drama
Rat named Darcy, A	0	F	F	F	F	F	V	F	familia
End of the Line	18	V	F	F	V	V	F	V	drama
Shiny Things, The	13	V	F	F	F	F	F	F	drama
Take it Back	18	F	V	F	F	F	F	F	comédia
Shark Bait	0	F	F	F	F	F	V	F	familia
Angry Pirate	13	F	V	F	F	F	F	V	comédia
Potentially Habitable Planet	13	F	V	F	F	V	F	F	comédia

À medida que todos os V/F dos títulos de filmes passarem pela varredura do comando CASE, o sistema SQL procura pelo primeiro 'V' para definir a categoria de cada um dos filmes.

Aqui está o que acontece quando Big Adventure passa pelo código:

```
UPDATE tabela_filme
SET categoria =
CASE
    WHEN drama = 'V' THEN 'drama' ← FALSO, nenhuma categoria
    WHEN comedia = 'V' THEN 'comédia' ← FALSO, nenhuma categoria
    WHEN acao = 'V' THEN 'ação' ← FALSO, nenhuma categoria
    WHEN horror = 'V' THEN 'terror' ← FALSO, nenhuma categoria
    WHEN ficcao = 'V' THEN 'ficção' ← FALSO, nenhuma categoria
    WHEN infantil = 'V' THEN 'família' ← FALSO, nenhuma categoria
    WHEN desenho = 'V' THEN 'família' ← FALSO, nenhuma categoria
    ELSE 'diversos' ← VERDADEIRO: Categoria definida como família e nós pulamos para o final e saímos da edição de código.
END;
```

Vamos fazer um com várias condições. Novamente, estamos procurando pelo primeiro valor 'V' aqui para definir a categoria.

Aqui está o que acontece quando 'Paraskavedekatriaphobia' é executado pelo código:

```
UPDATE tabela_filme
SET categoria =
CASE
    WHEN drama = 'V' THEN 'drama' ← VERDADEIRO: categoria definida como DRAMA; nós pulamos para o END (final) e saímos do código. Todos nossos outros valores V são ignorados.
    WHEN comedia = 'V' THEN 'comédia'
    WHEN acao = 'V' THEN 'ação'
    WHEN horror = 'V' THEN 'terror'
    WHEN ficcao = 'V' THEN 'ficção'
    WHEN infantil = 'V' THEN 'família'
    WHEN desenho = 'V' THEN 'família'
    ELSE 'diversos'
```

## Hácece que temos um problema

Vamos ter uns problemas. 'Great Adventure' é um desenho com censura de 18 anos. De alguma forma ele acabou sendo legalizado como 'família'.

### MENSAGEM

Data: Hoje Hora: 13:41

To: O Chefe

### ENQUANTO VOCÊ ESTAVA FORA

*Cliente muito nervoso.*

Telefonou	<input checked="" type="checkbox"/> Favor ligar	<input checked="" type="checkbox"/>
Ligou para te encontrar	<input type="checkbox"/> Ligará novamente	<input type="checkbox"/>
Quer ver você	<input type="checkbox"/> Retornou sua chamada	<input type="checkbox"/>

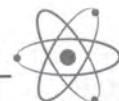
**MENSAGEM** *Uma senhora ligou para se queixar que seu filho assistiu um desenho com muita profanidade, agora ele vive perseguindo sua irmã menor e chamando-a de a P\*\*\*!@*

Recebida por: Mim URGENTE



## Aponte seu lápis

Altere a expressão CASE para fazer com que os desenhos sejam armazenados na categoria diversos e não 'família'.



# O PODER DO CÉREBRO

Qual a possibilidade de usarmos a censura 18 para evitarmos que isso aconteça no futuro?



## Aponte seu lápis

## Solução

Altere a expressão CASE para fazer com que os desenhos sejam armazenados na categoria diversos e não 'família'.

```

UPD@TE tabela_filme
SET categoria =
CASE
WHEN drama = 'V' THEN 'drama'
WHEN comedia = 'V' THEN 'comédia'
WHEN acao = 'V' THEN 'ação'
WHEN horror = 'V' THEN 'terror'
WHEN ficcao = 'V' THEN 'ficção'
WHEN infantil = 'V' THEN 'família'
WHEN desenho = 'V' AND censura = 0 THEN 'família'
ELSE 'diversos'
END;

```

Sua condição pode ter partes múltiplas adicionadas por um AND para seu WHEN para testar se o filme é um desenho e é censurado O. Se for, então ele é da categoria família.

# Perguntas Idiotas

P: Eu preciso usar ELSE?

R. É opcional. Se você simplesmente omitir esta linha e não precisar dela, mas é bom ter que atualizar os valores da sua tabela quando nada mais se ajusta à condição. É melhor ter um valor qualquer que NULL, por exemplo.

P: O que acontece se eu omitir o ELSE e nenhuma das condições WHEN forem cabíveis?

R: Nenhum dos valores será alterado na coluna que estiver atualizando.

P: E se eu quiser utilizar a expressão CASE em apenas algumas colunas e não em todas? Por exemplo, se eu quiser utilizar a expressão apenas onde minhas categorias sejam = 'diversos'. Posso utilizar o WHERE?

R: Sim, você pode utilizar uma cláusula WHERE depois da sua palavra-chave END. A expressão CASE só se aplicará para aquelas colunas que tenha delimitado pelo WHERE.

**P:** Posso utilizar a expressão CASE com qualquer outra coisa além de comandos UPDATE?

R: Sim, você pode utilizar uma expressão CASE com SELECT, INSERT, DELETE e, como viu, com UPDATE.

## **CONSTRUÇÃO COM CASE**

Seu chefe, que não tem opinião própria, decidiu alterar um pouco as coisas. Leia o email dele e escreva um só comando SQL que alcançará o que ele deseja.

Acontece que as novas categorias estão fazendo com que os consumidores passem por maus momentos tentando encontrar os filmes. Escreva um comando que elimine as categorias de censura 18 que você acabou de criar.

Direção: Equipe do Dataville Video

Para. 2

Assunto: Novas prateleiras significam novas categorias!

Oi galera,

Decidi criar mais algumas seções. Estou pensando que talvez os filmes censurados em 18 anos seja armazenados em prateleiras diferentes que os de censura 0 ou 13 anos. Vamos então criar 5 novas categorias:

terror-18

ação-18

drama-18

comédia-18

ficção-18

E se tiver algum filme qualquer com censura 0 na seção diversos, move-o para a prateleira da categoria Família.

Obrigado, isto vai ficar ótimo.

O chefe

Finalmente, delete todas aquelas colunas V/F que não precisamos mais.

## CONSTRUÇÃO COM CASE

Seu chefe, que não tem opinião própria, decidiu alterar um pouco as coisas. Leia o email dele e escreva um só comando SQL que alcançará o que ele deseja.

```
UPDATE tabela_filme
```

```
SET categoria =
```

```
CASE
```

```
WHEN drama = 'V' AND censura = '18' THEN 'drama-18'  
WHEN comedia = 'V' AND censura = '18' THEN 'comédia-18'  
WHEN acao = 'V' AND censura = '18' THEN 'ação-18'  
WHEN horror = 'V' AND censura = '18' THEN 'terror-18'  
WHEN ficcao = 'V' AND censura = '18' THEN 'ficção-18'  
WHEN diversos = 'V' AND censura = 'D' THEN 'família'  
END;
```

Acontece que as novas categorias estão fazendo com que os consumidores passem por maus momentos tentando encontrar os filmes. Escreva um comando que elimine as categorias de censura 18 que você acabou de criar.

```
UPDATE tabela_filme
```

```
SET categoria =
```

```
CASE
```

```
WHEN categoria = 'drama-18' THEN 'drama'  
WHEN categoria = 'comédia-18' THEN 'comédia'  
WHEN categoria = 'ação-18' THEN 'ação'  
WHEN categoria = 'terror-18' THEN 'terror'  
WHEN categoria = 'ficção-18' THEN 'ficção'  
END;
```

Para: Equipe do Dataville Video

De: Chefe

Assunto: Novas prateleiras significam novas categorias!

Oi galera,

Decidi criar mais algumas seções. Estou pensando que talvez os filmes censurados em 18 anos seja armazenados em prateleiras diferentes que os de censura 0 ou 13 anos. Vamos então criar 5 novas categorias:

terror-18

ação-18

drama-18

comédia-18

ficção-18

E se tiver algum filme qualquer com censura 0 na seção diversos, mova-o para a prateleira da categoria Família.

Obrigado, isto vai ficar ótimo.

O chefe

Finalmente, delete todas aquelas colunas V/F que não precisamos mais.

```
ALTER TABLE tabela_filme
```

```
DROP COLUMN drama;
```

```
DROP COLUMN comedia;
```

```
DROP COLUMN acao;
```

```
DROP COLUMN horror;
```

```
DROP COLUMN ficcao;
```

```
DROP COLUMN infantil;
```

```
DROP COLUMN desenho;
```

Quando um tabela. Não os filmes, m prateleiras e nele indicam filmes de categoria

Nós sabemos em uma cat alguma forn

## As tabelas podem ficar bem bagunçadas

Quando um filme chega à loja, ele é adicionado à tabela e torna-se a linha mais nova na tabela. Não há ordem para os filmes na sua tabela de filmes. E agora é hora de reorganizar os filmes, mas temos um pequeno problema. Nós sabemos que em cada uma das categorias cabe 20 filmes, e cada um dos mais de 3000 filmes tem que ter uma etiqueta indicando a categoria na qual ele pertence. **Nós precisamos selecionar os filmes de cada categoria em ordem alfabética dentro de sua própria categoria.**

Sabemos como realizar consultas no banco de dados para encontrar todos os filmes de uma categoria, mas precisamos deles listados em ordem alfabética na sua categoria, de alguma forma.

tabela\_filme

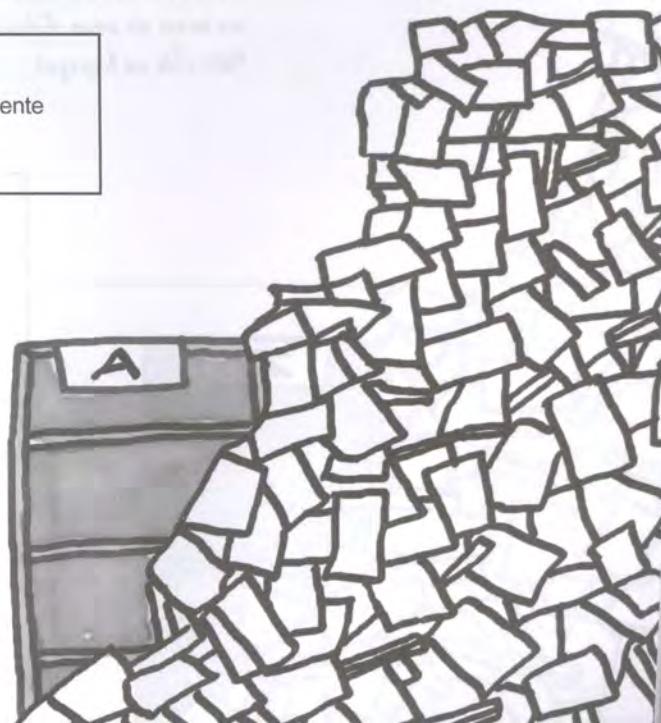
id_filme	titulo	censura	categoria	data_compra
84	Greg: The Untold Story	13	ação	5-2-2001
85	Mad Clowns	18	terror	20-11-1999
86	Paraskavedekatriaphobia	18	ação	19-4-2003
87	Rat named Darcy, A	0	familia	19-4-2003
88	End of the Line	18	diversos	5-2-2001
89	Shiny Things, The	13	drama	6-3-2002
90	Take it Back	18	comédia	5-2-2001
91	Shark Bait	0	diversos	20-11-1999
92	Angry Pirate	13	diversos	19-4-2003
93	Potentially Habitable Planet	13	ficção	5-2-2001
93	Potentially Habitable Planet	PG	scifi	2-5-2001
94	Cow-Gop-Wild	13	terror	3-2-2001

Estes são apenas alguns dos 3000 filmes



### Musculação cerebral

Como você organizaria estes dados alfabeticamente com um comando SQL?



# Nós precisamos de uma maneira de organizar os dados que selecionamos

Cada um dos mais de 3000 filmes deve ter uma etiqueta afixada indicando sua categoria. Então, ele deverá ser colocado na prateleira em ordem alfabética.

**Precisamos de uma lista principal dos filmes em ordem alfabética por título e categoria.** Até então, sabemos como utilizar o SELECT. Nós podemos facilmente selecionar os filmes por categoria, e podemos ainda selecionar filmes pela primeira letra do título e por sua categoria.

Mas para organizar nossa grande lista de filmes quer dizer que teríamos que escrever pelo menos 182 comandos SELECT, aqui estão alguns deles.

SELECT

```
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'A%' AND categoria = 'família'  
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'B%' AND categoria = 'família'  
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'C%' AND categoria = 'família'  
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'D%' AND categoria = 'família'  
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'E%' AND categoria = 'família'  
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'F%' AND categoria = 'família'  
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'G%' AND categoria = 'família'
```



Precisamos saber o título para que possamos cavucar nessa pilha de dados para achá-lo, bem como a categoria para imprimirmos a etiqueta e guardá-la na prateleira.

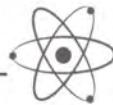


Esta é a letra do alfabeto com que o título do filme deve começar.



Essa é a categoria que estamos procurando.

Há 182 consultar porque temos 7 categorias e 26 letras no alfabeto. Este número não inclui filmes que tenham números no início de seus títulos (como 101 Dálmatas, ou 2001: Uma Odisséia no Espaço).



## PODER DO CÉREBRO

Onde você acha que os filmes com títulos que começam com letra ou um outro caractere que não seja letra – como um ponto de exclamação – aparecerão na lista?



 Aponte seu lápis —

Ainda temos que colocar os títulos em ordem alfabética, manualmente, dentro de suas próprias categorias utilizando as letras que seguem após a inicial 'A' para ficarem em ordem.

Preste bem atenção para alguns dos resultados de apenas uma das nossas 182 (ou mais) consultas. Tente colocá-los em ordem manualmente.

```
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'A%' AND categoria = 'família';
```

Um pedaço do resultado de nossa pesquisa.

titulo	categoria
Airplanes and Helicopters	familia
Are You Paying Attention?	familia
Acting Up	familia
Are You My Mother?	familia
Andy Sighs	familia
After the Clowns Leave	familia
Art for Kids	familia
Animal Adventure	familia
Animal Crackerz	familia
Another March of the Penguins	familia
Anyone Can Grow Up	familia
Aaargh!	familia
Aardvarks Gone Wild	familia
Alaska: Land of Salmon	familia
Angels	familia
Ann Eats Worms	familia
Awesome Adventure	familia
Annoying Adults	familia
Alex Needs a Bath	familia
Aaargh! 2	familia



## Aponte seu lápis Solução

Ainda temos que colocar os títulos em ordem alfabética, manualmente, dentro de suas próprias categorias utilizando as letras que seguem após a inicial 'A' para ficarem em ordem.

Preste bem atenção para alguns dos resultados de apenas uma das nossas 182 (ou mais) consultas. Tente colocá-los em ordem manualmente.

```
SELECT titulo, categoria FROM tabela_filme WHERE titulo LIKE 'A%' AND categoria = 'familia';
```

titulo	categoria
Aaargh!	familia
Aaargh! 2	familia
Aardvarks Gone Wild	familia
Acting Up	familia
After the Clowns Leave	familia
Airplanes and Helicopters	familia
Alaska: Land of Salmon	familia
Alex Needs a Bath	familia
Andy Sighs	familia
Angels	familia
Animal Adventure	familia
Animal Crackerz	familia
Ann Eats Worms	familia
Annoying Adults	familia
Another March of the Penguins	familia
Anyone Can Grow Up	familia
Are You My Mother?	familia
Are You Paying Attention?	familia
Art for Kids	familia
Awesome Adventure	familia

**Quanto tempo demorou para ordenar estes 20 filmes?**

**Você pode imaginar quanto tempo duraria para ordenar 3000 ou mais filmes desta forma?**

Os títulos iniciando Are you..., vão para o final da lista, uma vez que a letra após a letra inicial A é um r, mas ai temos que olhar na sétima letra do título antes de imaginarmos aonde cada um destes deveria ser armazenado.

## Ente um simples ORDER BY

Ve disse que precisa ordenar sua tabela? Bem, acontece que pode usar seu Sistema SQL para SELECT (selecionar) algo e ORDER (ordenar) os dados que ele exibir como resultado como sendo uma nova coluna da tabela.

*Não há surpresas aqui.  
Isto é exatamente  
como o comando SELECT  
que acabamos de utilizar.*

*Aqui está o novo pedaço.  
Ele faz realmente o que  
ele quer dizer. Ele diz  
ao Sistema para exibir  
os dados em ordem  
alfabética pelo título.*

```
SELECT titulo, categoria
FROM tabela_filme
WHERE
    titulo LIKE 'A%'
AND
    categoria = 'família'
ORDER BY titulo;
```

Sério. Você está dizendo  
que este é o único modo para  
colocar nossos dados em ordem alfabética?  
Não HÁ POSSIBILIDADE de eu fazer isso  
para cada letra do alfabeto.



Aponte seu lápis

Você está certa. O que podemos  
retirar da tabela acima para fazê-la  
muito mais poderosa?

Pare! Faça este exercício antes de virar a página.

## ORDER (Ordene) uma só coluna

Se sua consulta utilizar ORDER BY `título`, não precisamos pesquisar pelos títulos que comecem por uma letra em particular porque a consulta exibe os dados listados em ordem alfabética por título.

Tudo que precisamos fazer é tirar a parte do código que diz `título LIKE`, e ORDER BY `título` fará o resto.



Aponte seu lápis

O que nós podemos retirar da tabela acima para fazê-la muito mais poderosa?

```
SELECT titulo, categoria
FROM tabela_filme
WHERE
titulo LIKE 'A%
AND
categoria = 'família'
ORDER BY titulo;
```



```
SELECT titulo, categoria
FROM tabela_filme
WHERE
categoria = 'família'
ORDER BY titulo;
```

Desta vez trabalharemos na lista inteira de filmes da categoria família.

Melhor ainda, esta lista vai incluir os filmes no qual os títulos começem com números. Eles serão os primeiros na lista.

Este não é o final dos resultados; não temos espaço suficiente para exibi-los todos aqui. Eles continuam até os títulos que começam pela letra Z.

ORDER BY permite que você ordene alfabeticamente qualquer coluna.

Perceba que os primeiros títulos começam com um número.

titulo	categoria
1 Crazy Alien	família
10 Big Bugs	família
101 Alsatians	família
13th Birthday Magic	família
2 + 2 is 5	família
3001 Ways to Fall	família
5th Grade Girls are Evil	família
7 Year Twitch	família
8 Arms are Better than 2	família
Aaargh!	família
Aaargh! 2	família
Aardvarks Gone Wild	família
Acting Up	família
After the Clowns Leave	família
Airplanes and Helicopters	família
Alaska: Land of Salmon	família
Alex Needs a Bath	família
Andy Sighs	família
Angels	família
Animal Adventure	família
Animal Crackerz	família
Ann Eats Worms	família
Annoying Adults	família
Another March of the Penguins	família
Anyone Can Grow Up	família
Are You My Mother?	família
Are You Paying Attention	família
Art for Kids	família
Awesome Adventure	família



## Exercícios

Crie uma tabela simples com uma só coluna do tipo CHAR (1) chamada 'teste\_chars'.

Insira números, letras (maiúsculas e minúsculas), e caracteres não-alfabéticos exibidos abaixo nesta coluna, cada um em uma linha separada. Insira um espaço e deixe uma linha NULL.

Experimente sua nova consulta ORDER BY na coluna e preencha as lacunas no livro de *Regras de ordem do SQL* exibido abaixo.

0123ABCDabcd!@#\$%^&\*()\_-  
+=[]{};:'"\|`~,.<>/?

## Regras de ordem do SQL

Após executar sua consulta ORDER BY, preencha as lacunas utilizando a ordem que os caracteres aparecem no seu resultado para ajudar.

Caracteres não-alfabéticos apareceram ..... dos números.

Números apareceram ..... dos caracteres de texto.

Valores NULL apareceram ..... dos números.

Valores NULL apareceram ..... dos textos alfabéticos.

Caracteres em maiúsculo apareceram ..... que os minúsculos.

"A 1" aparecerá ..... de "A1".

## Regras de ordem do SQL

Após executar sua consulta ORDER BY, coloque estes caracteres na ordem em que eles apareceram no resultado da consulta.

+ = ! ( & ~ "  
\* @ ? ' <

Lembra-se como inserir uma aspa simples? Elas são complicadas.





Crie uma tabela simples com uma só coluna do tipo CHAR (1) chamada 'teste\_chars'.

Insira números, letras (maiúsculas e minúsculas), e caracteres não-alfabéticos exibidos abaixo nesta coluna, cada um em uma linha separada. Insira um espaço e deixe uma linha NULL.

Experimente sua nova consulta ORDER BY na coluna e preencha as lacunas no livro de Regras de ordem do SQL exibido abaixo.

```
!"#$%&'()*+,-./0123:;<=>
?@ABCD[\]^_`abcd{|}~
```

A ordem que os caracteres devem ser exibidos em sua consulta. Observe o espaço no início. Sua ordem pode ter um pouco diferente dependendo do seu Sistema SQL. O objetivo aqui é mostrar que há uma ordem, e qual é esta ordem para seu Sistema SQL.

### Regras de ordem do SQL

Após executar sua consulta ORDER BY, preencha as lacunas utilizando a ordem que os caracteres aparecem no seu resultado para ajudar.

Caracteres não-alfabéticos apareceram antes e depois dos números.

Números apareceram antes dos caracteres de texto.

Valores NULL apareceram antes dos números.

Valores NULL apareceram antes dos textos alfabéticos.

Caracteres em maiúsculo apareceram antes que os minúsculos.

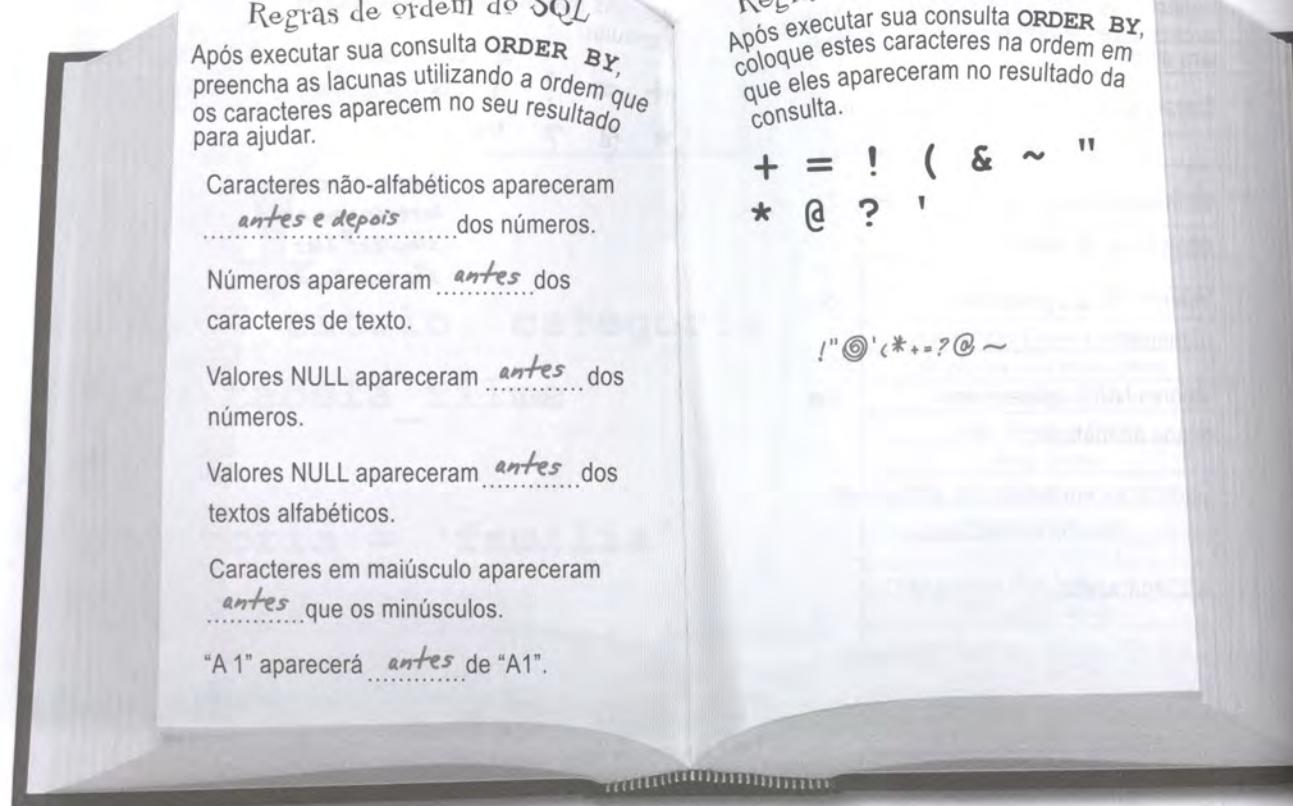
"A 1" aparecerá antes de "A1".

### Regras de ordem do SQL

Após executar sua consulta ORDER BY, coloque estes caracteres na ordem em que eles apareceram no resultado da consulta.

+ = ! ( & ~ "  
\* @ ? '

!"@`\*+=?@~



## ORDER com duas

### colunas

vez que está tudo sob controle. podemos ordenar em ordem alfabética os filmes, e podemos criar listas separadas para cada categoria.

talmente seu chefe tem algo mais pra você fazer...

mente, você pode ordenar múltiplas colunas no mesmo comando.

Para: Equipe do Dataville Video

De: Chefe

Assunto: Fora com os filmes antigos

Oi,

Acho que precisamos eliminar alguns dos filmes que nós temos por mais tempo. Você pode vir neste final de semana e expedir uma lista de filmes em cada categoria por ordem de data de compra?

Isto seria ótimo.

O chefe

*Queremos ter certeza que a coluna data\_compra será exibida nos resultados.*

```
SELECT titulo, categoria, data_compra
FROM tabela_filme
ORDER BY categoria, data_compra;
```

↑  
Esta será a primeira coluna a ser ordenada. Nós iremos obter uma lista de cada filme na loja separado por categoria.

↑  
E esta será a segunda coluna ordenada, depois que a coluna categoria já tiver sido ordenada.



### Musculação cerebral

Os filmes mais antigos serão exibidos em primeiro ou em último em cada categoria? E o que você acha que acontecerá se dois estiverem na mesma categoria com a mesma data de compra? Qual ele exibirá primeiro.

## ORDER com colunas múltiplas

Você não está restrito a procurar em apenas duas colunas. Você pode pesquisar por tantas colunas quanto precisar.

Observe este ORDER BY com três colunas. Está acontecendo o que e como as tabelas têm sido ordenadas.

```
SELECT * FROM tabela_filme
ORDER BY categoria, data_compra, titulo;
```

Você pode pesquisar pelo tanto de colunas que precisar.

Primeiro seus dados são ordenados por categoria, uma vez que esta foi a primeira coluna listada através do ORDER BY. Os resultados estão listados de A a Z.

Depois, os resultados (cada categoria na tabela começando com A já que é assim que as categorias estão ordenadas agora) são separados por data, com a data mais antiga vindo primeiro. Datas são sempre separadas por ano, depois mês e então, por dia.

## Categorias

começando com

A ----->

Z ----->

## Data de compra primeiro os mais antigos

20-11-1999 ----->

19-4-2003 ----->

## Data de compra primeiro os mais antigos

20-11-1999 ----->

Títulos  
Começando  
com

A  
|  
Z

Títulos  
Começando  
com

A  
|  
Z

Títulos  
Começando  
com

A

Finalmente, os resultados categoria começando com que agora é ordenada por data da compra são ordenados título, novamente, começando pelo A e terminando no Z.

Nossa ta  
original f

De fato, não  
nenhuma ord  
os filmes só  
na ordem em  
registros t  
inseridos ne

E os

## ma tabela\_filme organizada

ver o que este comando SELECT exibe quando o executamos em nossa tabela original.

*Lados (cada um A, por data idos por negando Z).*

*Nessa tabela visual tabela\_filme.*

*não, não há ordem aqui, os filmes só aparecem dentro em que os outros foram inseridos na tabela.*

id_filme	titulo	censura	categoria	data_compra
84	Greg: The Untold Story	13	ação	5-2-2001
85	Mad Clowns	18	terror	20-11-1999
86	Paraskavedekatriaphobia	18	ação	19-4-2003
87	Rat named Darcy, A	0	família	19-4-2003
88	End of the Line	18	diversos	5-2-2001
89	Shiny Things, The	13	drama	6-3-2002
90	Take it Back	18	comédia	5-2-2001
91	Shark Bait	0	diversos	20-11-1999
92	Angry Pirate	13	diversos	19-4-2003
93	Potentially Habitable Planet	13	ficção	5-2-2001
94	Cows Gone Wild	18	terror	19-3-2007
94	Cows Gone Wild	18	terror	3-1-2007

E os dados ordenados da nossa consulta:

Terceira coluna ordenada.

Primeira coluna ordenada.

Segunda coluna ordenada.

id_filme	titulo	censura	categoria	data_compra
84	Greg: The Untold Story	13	ação	5-2-2001
86	Paraskavedekatriaphobia	18	ação	19-4-2003
90	Take it Back	18	comédia	5-2-2001
89	Shiny Things, The	13	drama	6-3-2002
83	Bobby's Adventure	0	família	6-3-2002
87	Rat named Darcy, A	0	família	19-4-2003
85	Mad Clowns	18	terror	20-11-1999
91	Shark Bait	0	diversos	20-11-1999
88	End of the Line	18	diversos	5-2-2001
93	Potentially Habitable Planet	13	ficção	5-2-2001



Não gosto de filmes antigos. E se eu quiser ver os filmes novos primeiro? Tenho que ler a lista de baixo pra cima?

**SQL tem uma palavra-chave que inverte a ordem.**

Por padrão, SQL exibe as colunas (ORDER BY) em ordem CRESCENTE (ASCENDENT). Isto quer dizer que você sempre obterá resultados de A para Z e de 1 para 99.999. Se preferir que a ordem seja invertida, vai querer que os dados fiquem em ordem DECRESCENTE (DESCENDENT). Você pode utilizar a palavra-chave DESC logo após o nome da coluna.

---

não existem  
Perguntas Idiotas

---

**P:** Eu pensei que DESC servisse para obter a DESCRIÇÃO de uma tabela. Você tem certeza que ele vai funcionar para inverter o comando ORDER?

**R:** Sim. Tem tudo a ver com contexto. Quando o utilizar na frente do nome de uma tabela – por exemplo, DESC tabela\_filme; - terá a descrição da tabela. Neste caso, DESC é a abreviação para DESCRIBE.

Quando você o utiliza em uma cláusula ORDER, ele vai significar DESCENDENTE, e é assim que ele vai ordenar os resultados.

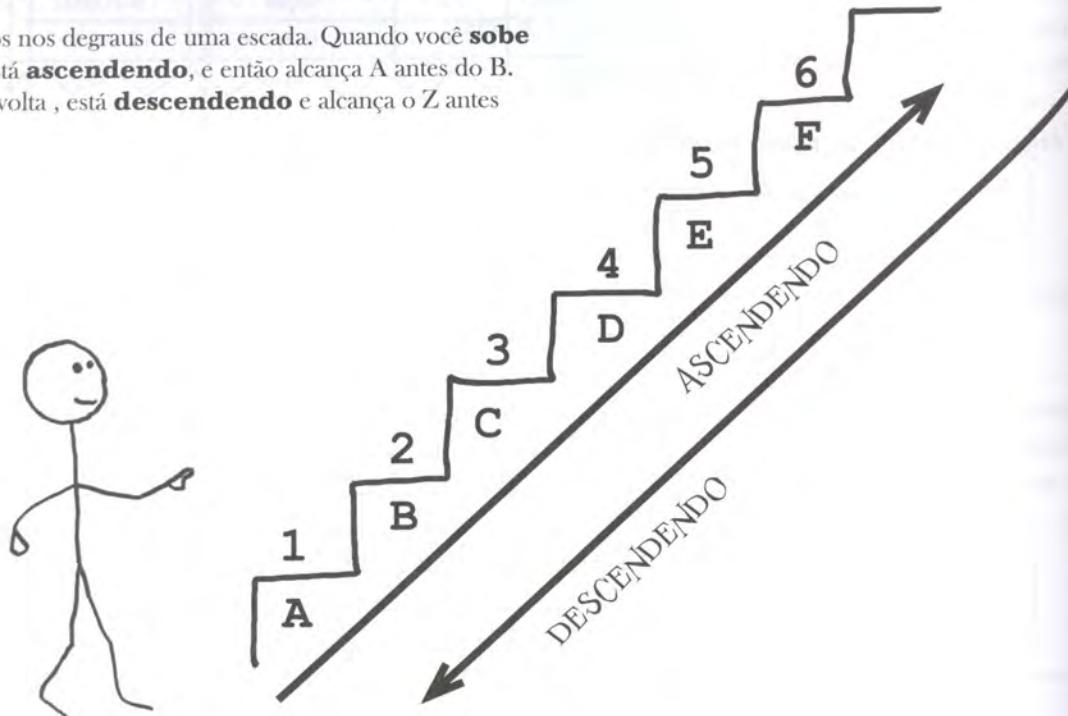
**P:** Eu posso utilizar as palavras completas DESCRIBE e DESCENDING na minha consulta para evitar confusão?

**R:** Você pode fazer isso com DESCRIBE, mas DESCENDING não vai funcionar.

Utilize a palavra-chave DESC depois do nome da sua coluna nas cláusulas ORDER BY para inverter a ordem de seus resultados

## Inverta o comando ORDER com DESC

Imagine seus dados nos degraus de uma escada. Quando você **sobe** as escadas, você está **ascendendo**, e então alcança A antes do B. Quando desce de volta, está **descendendo** e alcança o Z antes do A.



Esta consulta nos dá uma lista dos filmes ordenados pela data da compra, com os **mais novos**, primeiro. Para cada data, os filmes comprados naquela data, estarão em ordem alfabética.

```
SELECT titulo, data_compra
FROM tabela_filme
ORDER BY titulo ASC, data_compra DESC;
```

Podemos utilizar o ASC aqui, mas não é necessário. Apenas lembre-se que ASC é a ordem padrão.

Se quisermos ordenar nossos dados de Z para A ou de 9 para 1, temos que utilizar a palavra-chave DESC.

Para: Equipe do Dataville Video

De: Chefe

Assunto: Brindes por todo lado!

Oi galera,

A loja está com uma cara ótima! Vocês classificaram toda aquela pilha de filmes nos seus devidos lugares e, graças àquelas cláusulas ORDER BY nas SQLs de vocês, todos podem encontrar exatamente o que estão procurando.

Para recompensá-los por todo o trabalho árduo, darei uma festa com pizza na minha casa hoje à noite. Apareçam às 6.

Não se esqueçam de levar aqueles relatórios.

O chefe

P.S.: Não vista nada muito chique, tenho algumas prateleiras que estou me coçando para organizá-las...

# O problema da líder de vendas de biscoitos da Bandeirantes

A líder da tropa de Tropa de Garotas Bandeirantes está tentando computar qual das garotas vendeu mais biscoitos. Até agora, ela tem uma tabela das vendas das garotas para cada dia.

Eu preciso encontrar a vencedora logo. Ninguém gosta de uma Bandeirante nervosa.



Edwina, a confusa líder da Tropa de Garotas Bandeirantes

Bandeirante que realizou a venda.

venda\_biscoito

Quantidade em dólares dos valores arrecadados.

Venda desti

id	primeiro_nome	venda	data_venda
1	Lindsay	32.02	6-3-2007
2	Paris	26.53	6-3-2007
3	Britney	11.25	6-3-2007
4	Nicole	18.96	6-3-2007
5	Lindsay	9.16	7-3-2007
6	Paris	1.52	7-3-2007
7	Britney	43.21	7-3-2007
8	Nicole	8.05	7-3-2007
9	Lindsay	17.62	8-3-2007
10	Paris	24.19	8-3-2007
11	Britney	3.40	8-3-2007
12	Nicole	15.21	8-3-2007
13	Lindsay	0	9-3-2007
14	Paris	31.99	9-3-2007
15	Britney	2.58	9-3-2007
16	Nicole	0	9-3-2007
17	Lindsay	2.34	10-3-2007
18	Paris	13.44	10-3-2007
19	Britney	8.78	10-3-2007
20	Nicole	26.82	10-3-2007
21	Lindsay	3.71	11-3-2007
22	Paris	0.56	11-3-2007
23	Britney	34.19	11-3-2007
24	Nicole	7.77	11-3-2007
25	Lindsay	16.23	12-3-2007
26	Paris	0	12-3-2007
27	Britney	4.50	12-3-2007
28	Nicole	19.22	12-3-2007



Aponte seu lápis

A Bandeirante com o maior total vendido ganhará aulas grátis de hipismo. Todas as Bandeirantes querem vencer, então é crucial que Edwina descubra a vencedora correta antes das coisas ficarem feias.

Utilize seu talento com a cláusula ORDER BY para escrever uma consulta que ajudará Edwina a encontrar o nome da vencedora.

## Aponte seu lápis Solução

A Bandeirante com o maior total vendido ganhará aulas grátis de hipismo. Todas as Bandeirantes querem vencer, então é crucial que Edwina descubra a vencedora correta antes das coisas ficarem feias.

Utilize seu talento com a cláusula ORDER BY para escrever uma consulta que ajudará Edwina a encontrar o nome da vencedora.

```
SELECT primeiro_nome, venda
FROM venda_biscoito
ORDER BY primeiro_nome;
```

Aqui está nossa consulta...

...e aqui estão nossos resultados.

primeiro_nome	vendas
Nicole	19.22
Nicole	0.00
Nicole	8.05
Nicole	26.82
Nicole	7.77
Nicole	15.21
Nicole	18.96
Britney	3.40
Britney	2.58
Britney	4.50
Britney	11.25
Britney	8.78
Britney	43.21
Britney	34.19
Lindsay	17.62
Lindsay	9.16
Lindsay	0.00
Lindsay	32.02
Lindsay	2.34
Lindsay	3.71
Lindsay	16.23
Paris	26.53
Paris	0.00
Paris	0.56
Paris	1.52
Paris	13.44
Paris	24.19
Paris	31.99

96.03

107.91

81.08

98.23

As vendas para cada garota ainda devem ser somadas manualmente para encontrar a vencedora.

# SUM pode somá-las para nós

As apostas estão altas. Não podemos cometer um erro e arriscar a fazer nossas Garotas Bandeirantes nervosas. Ao invés de somá-las nós mesmos, podemos fazer com que o SQL faça o trabalho pesado pela gente.

A linguagem SQL possui algumas palavras-chave chamadas de *funções*. Funções são pedaços de códigos que realizam operações em um valor ou valores. O primeiro que mostraremos realiza operações matemáticas em uma coluna. **Nós utilizaremos a função SUM que funciona para somar** os valores de uma coluna designada através dos parênteses. Vamosvê-la em ação.

*A função SUM totaliza o valor na coluna venda.*

*SUM é conhecida como uma função. Isto quer dizer que ela executa uma ação na coluna perto dela que esteja entre parênteses.*

**SELECT SUM(venda)  
FROM venda\_biscoito  
WHERE primeiro\_nome = 'Nicole' ;**

*Isto restringe a consulta para adicionar somente as vendas da Nicole. Caso contrário ela estaria totalizando a coluna venda por completo.*

```
File Edit Window Help TheWinnerls
> SELECT SUM(venda) FROM venda_biscoito
-> WHERE primeiro_nome = 'Nicole';
+-----+
| SUM (venda) |
+-----+
|      96.03  |
+-----+
1 row in set (0.00 sec)
```

**Agora precisamos dos outros três totais e estaremos prontos. Mas seria mais fácil se pudéssemos fazer isso em uma só consulta...**



Tente em casa

Tente você mesmo. Crie uma tabela como a venda\_biscoito e insira alguns valores decimais nela. Então, trabalhe nas consultas que você encontrará nas próximas páginas.

Exercícios

## SUM (some) todas de uma vez com GROUP BY

uma maneira de SUM (somar) a venda de cada uma das garotas ao mesmo tempo. Nós teremos que adicionar um GROUP BY no nosso comando SUM. Este comando agrupa todos os primeiros nomes para cada garota e o total de vendas por grupo.

```

    ↗ SUM (soma)
SELECT primeiro_nome, SUM (venda)      todas as colunas.
FROM venda_biscoito
GROUP BY primeiro_nome                  ← Agrupa todos os valores primeiro_nome.
ORDER BY SUM (venda) DESC;              ↗ Queremos os valores exibidos de maior
                                         para o menor para que possamos ver a
                                         vencedora mais facilmente.

Temos que ordenar pela mesma
soma que nós selecionamos.
                                         ← Este comando totaliza todas
                                         as colunas venda em cada grupo
                                         primeiro_nome.
  
```

primeiro_nome	venda
Nicole	19.22
Nicole	0.00
Nicole	8.05
Nicole	26.82
Nicole	7.77
Nicole	15.21
Nicole	18.96

primeiro_nome	venda
Paris	26.53
Paris	0.00
Paris	0.56
Paris	1.52
Paris	13.44
Paris	24.19
Paris	31.99

primeiro_nome	venda
Lindsay	17.62
Lindsay	9.16
Lindsay	0.00
Lindsay	32.02
Lindsay	2.34
Lindsay	3.71
Lindsay	16.23

```

File Edit Window Help TheWinnerReallyIs
> SELECT primeiro_nome, SUM (venda)
-> FROM venda_biscoito GROUP BY primeiro_nome
-> ORDER BY SUM (venda) DESC;
+-----+-----+
| primeiro_nome | sum(venda) |
+-----+-----+
| Britney      |     107.91 |
| Paris        |      98.23 |
| Nicole       |      96.03 |
| Lindsay      |      81.08 |
+-----+-----+
4 rows in set (0.00 sec)

  
```

← vencedora  
→ Britney

# AVG combinada com GROUP BY

As outras garotas ficaram desapontadas, então Edwina decidiu dar outro prêmio para a garota com a maior média diária. Ela utilizou a função AVG.

Cada garota tem sete dias de vendas. Para cada garota a função somou suas vendas e então dividiu por 7.

*Novamente, nós estamos agrupando todos os valores da coluna primeiro\_nome...*

*...mas desta vez estamos obtendo a média dos valores.*

```
SELECT primeiro_nome, AVG (venda)
FROM venda_biscoito
GROUP BY primeiro_nome;
```

*AVG adiciona todos os valores em um grupo e então divide pelo número total de valores para encontrar o valor médio para aquele grupo.*

primeiro_nome	venda
Nicole	19.22
Nicole	0.00
Nicole	8.05
Nicole	26.82
Nicole	7.77
Nicole	15.21
Nicole	18.96

primeiro_nome	venda
Paris	26.53
Paris	0.00
Paris	0.56
Paris	1.52
Paris	13.44
Paris	24.19
Paris	31.99

primeiro_nome	venda
Lindsay	17.62
Lindsay	9.16
Lindsay	0.00
Lindsay	32.02
Lindsay	2.34
Lindsay	3.71
Lindsay	16.23

*Oh-Oh, Britney venceu de novo! Nós precisamos criar uma maneira para encontrar um lugar para o segundo vencedor.*

```
File Edit Window Help TheWinnerReallyIs
> SELECT primeiro_nome, AVG (venda)
-> FROM venda_biscoito GROUP BY primeiro_nome;
+-----+-----+
| primeiro_nome | AVG(venda) |
+-----+-----+
| Nicole       | 13.718571 |
| Britney      | 15.415714 |
| Lindsay      | 11.582857 |
| Paris        | 14.032857 |
+-----+-----+
4 rows in set (0.00 sec)
```

## MIN e MAX

Edwina deseja deixar nada de fora, Edwina dá uma rápida olhada para os valores MIN e MAX de sua tabela para ver se alguma das outras garotas teve a maior venda para um só dia, ou ainda se Britney teve um dia em que foi pior e ficou com o menor valor de venda para um dia em comparação com as outras...

Vamos utilizar a função MAX para encontrar **os maiores valores em uma coluna**. MIN nos dará os **menores valores em uma coluna**.

```
SELECT primeiro_nome, MAX (venda)
FROM venda_biscoito
GROUP BY primeiro_nome;
```

MAX retorna os valores mais altos de venda para cada garota.

primeiro_nome	venda
Nicole	26.82
Britney	43.21
Lindsay	32.02
Paris	31.99

Surpresa,  
Britney obteve  
a maior venda  
diária.

```
SELECT primeiro_nome, MIN (venda)
FROM venda_biscoito
GROUP BY primeiro_nome;
```

MIN retorna o único valor de venda menor para cada garota.

primeiro_nome	venda
Nicole	0.00
Britney	2.58
Lindsay	0.00
Paris	0.00

É engraçado que  
cada garota fez algo  
muito menor um dia,  
mesmo no pior dia da  
Britney ela conseguiu um  
pouco de dinheiro.

Isto está ficando sério.  
Talvez eu possa dar o prêmio para  
a garota que vendeu biscoitos por mais  
dias que qualquer uma das outras.

## COUNT (conte) os dias

Em descobrir qual garota vendeu biscoitos em mais dias que qualquer uma outra, Edwina tenta calcular por quantos dias os biscoitos foram vendidos com a função COUNT. COUNT exibirá o **número de linhas de uma coluna**.

```
SELECT COUNT (data_venda)
FROM venda_biscoito;
```

COUNT exibe o número de linhas na coluna data\_venda.  
Se o valor for NULL ele não é contado.





Aponte seu lápis

venda\_biscoito

id	primeiro_nome	venda	data_venda
1	Lindsay	32.02	3-6-2007
2	Paris	26.53	3-6-2007
3	Britney	11.25	3-6-2007
4	Nicole	18.96	3-6-2007
5	Lindsay	9.16	3-7-2007
6	Paris	1.52	3-7-2007
7	Britney	43.21	3-7-2007
8	Nicole	8.05	3-7-2007
9	Lindsay	17.62	3-8-2007
10	Paris	24.19	3-8-2007
11	Britney	3.40	3-8-2007
12	Nicole	15.21	3-8-2007
13	Lindsay	0	3-9-2007
14	Paris	31.99	3-9-2007
15	Britney	2.58	3-9-2007
16	Nicole	0	3-9-2007
17	Lindsay	2.34	3-10-2007
18	Paris	13.44	3-10-2007
19	Britney	8.78	3-10-2007
20	Nicole	26.82	3-10-2007
21	Lindsay	3.71	3-11-2007
22	Paris	.56	3-11-2007
23	Britney	34.19	3-11-2007
24	Nicole	7.77	3-11-2007
25	Lindsay	16.23	3-12-2007
26	Paris	0	3-12-2007
27	Britney	4.50	3-12-2007
28	Nicole	19.22	3-12-2007

Aqui está a tabela original. O que você acha que será exibido pela consulta?

.....

.....

SELECT

Primeiramen  
chave DIST

Este número representa realmente o número de dias que os biscoitos foram vendidos?

.....

.....

Escreva uma consulta que nos dará o número de dias que cada garota vendeu biscoitos.

.....

.....

Olhe pa  
não há n  
duplicida  
mante!



Aponte seu lápis

Solução

Aqui está a tabela original. O que você acha que será exibido pela consulta?

28 dias de vendas

Agora vamos

Este número representa realmente o número de dias que os biscoitos foram vendidos?

*Não. Este número simplesmente representa o número de valores na tabela para a coluna data\_venda*

Escreva uma consulta que nos dará o número de dias que cada garota vendeu biscoitos.

```
SELECT primeiro_nome, COUNT(data_venda)
FROM venda_biscoito
GROUP BY primeiro_nome;
```



Você poderia ter utilizado simplesmente um ORDER BY na coluna data\_venda e olhar para a última e a primeira data e contar quantos dias os biscoitos foram vendidos. Certo?

**Bem, não. Você não poderia ter certeza de que há dias faltando entre a primeira e a última data.**

De fato, há uma maneira muito mais fácil para descobrir os dias em que os biscoitos foram vendidos: utilizando a palavra-chave DISTINCT. Ela não só serve para fornecer a contagem que você estava precisando, mas também obtém uma lista das datas que não estão em duplicidade.

## SELECT DISTINCT (selecione diferentes) valores

...eramente, vamos dar uma olhada na palavra-chave DISTINCT **sem** a função COUNT.

```
SELECT DISTINCT data_venda
FROM venda_biscoito
ORDER BY data_venda;
```

Já que DISTINCT é uma palavra-chave e não uma função você não precisa de parênteses para a coluna data\_venda.

Aqui está nosso ORDER BY para que possamos ver a primeira e a última data de venda.

```
File Edit Window Help NoDuplicates
> SELECT DISTINCT data_venda
  FROM venda_biscoito
    -> ORDER BY data_venda;
+-----+
| data_venda |
+-----+
| 2007-03-06 |
| 2007-03-07 |
| 2007-03-08 |
| 2007-03-09 |
| 2007-03-10 |
| 2007-03-11 |
| 2007-03-12 |
+-----+
7 rows in set (0.00 sec)
```

...vamos tentar com a função COUNT:

```
SELECT COUNT (DISTINCT data_venda)
  FROM venda_biscoito;
```

Não precisamos de um ORDER BY porque COUNT irá exibir um só número. Não há o que ser ordenado aqui.



Musculação cerebral

Experimente esta consulta, e então utilize-a para descobrir qual garota vendeu biscoito por mais dias.

Resposta: Bruna

# Quem sou eu?



Uma porção de funções e palavras-chave com a fantasia completa estão jogando um jogo comum em festas, "Quem sou eu?" Eles darão uma pista – você tenta adivinhar quem são eles baseando-se no que eles dizem. Presuma que eles sempre falarão a verdade sobre si. Preencha as lacunas em branco à direita para identificar os participantes. Também, para cada participante, escreva se isso é uma função ou uma palavra-chave.

Participantes de hoje:

COUNT, DISTINCT, AVG, MIN, GROUP BY, SUM, MAX

Nome	Função ou palavra-chave
------	-------------------------

O resultado que você obtém por me usar pode não valer muito.

O que eu ponho pra fora é maior do que qualquer coisa que eu ponho pra dentro.

Eu darei um resultado único.

Eu direi quantos haviam ali.

Você precisa me utilizar se quiser obter uma soma.

Eu estou interessado apenas nos grandes números.

Como eu sou? Algo mediano.

---

## Pnão existemerguntas Idiotas

---

**P:** Já que estava procurando pelos maiores valores com AVG, MAX e MIN, você não poderia ter utilizado uma cláusula ORDER BY?

**R:** Sim poderia, e teria sido uma ótima idéia. Optamos por não fazê-lo para que não houvesse interferência nas consultas e facilitar seu aprendizado de novas funções. Dê uma olhada para trás nas funções e visualize ali o ORDER BY. Notou como os resultados se alterariam?

**P:** Aquela palavra-chave DISTINCT parece bem útil. Posso utilizá-la com qualquer coluna que eu queira?

**R:** Sim, pode. Ela é especialmente útil quando há múltiplos registros com o mesmo valor em uma só coluna, e você simplesmente quer ver a variedade dos valores, e não uma lista enorme com valores em duplicidade.

**P:** Fazer a consulta para MIN() não tinha nada a ver com Edwina encontrar um vencedor, tinha?

**R:** Não, mas teria tê-la ajudado a encontrar as meninas com os piores resultados. Ano que vem ela poderá observá-las de perto para motivá-las ainda mais.

**P:** Falando do MIN, o que acontece se há um valor NULL na coluna?

**R:** Boa pergunta. Nenhum NULL é retornado uma vez sequer por nenhuma destas funções, porque NULL é a ausência de um valor, não a mesma coisa que zero.

Respostas na página 225.

LIMIT

Agora irem  
consulta or

SE  
FRO  
GRO  
ORI



Já que temos  
ser ainda m  
poderíamos  
linhas quere

SEL  
FRO  
GRO  
ORD  
LIM

Enquanto te  
parece ajuda  
tenha uma li  
mas você qu  
aqueelas e nã



Hum. AVG, MAX e COUNT realmente não funcionaram de forma que determinasse o vencedor do segundo lugar. Eu imagino se posso utilizar SUM para descobrir qual garota obteve o segundo lugar e, assim, dar a ela um presente.



### PODER DO CÉREBRO

Imagine se não tivéssemos **quatro**, mas quarenta Garotas Bandeirantes. Como poderíamos utilizar o SUM para conseguir a segunda posição?

## LIMIT (limite) o número de resultados

Vamos utilizar SUM para determinar a segunda colocada. Vamos olhar para trás na consulta original e seus resultados para nos ajudar a descobrir como obter uma 2<sup>a</sup> vencedora.

```
SELECT primeiro_nome, SUM(venda)
FROM venda_biscoito
GROUP BY primeiro_nome
ORDER BY SUM(venda) DESC;
```

*É crucial que utilizemos ORDER BY aqui; de outra forma, nossos resultados seriam arbitrários.*

primeiro_nome	venda
Britney	107.91
Paris	98.23
Nicole	96.03
Lindsay	81.08

*Nós realmente queremos os dois primeiros resultados.*

*Paris é nossa vencedora na 2<sup>a</sup> posição!  
Nicole parou de falar com ela depois disso.*

Temos apenas quatro resultados, é fácil perceber quem ficou em segundo lugar. Mas se nós queremos ainda mais precisos, poderíamos limitar os resultados apenas para as duas primeiras garotas. Dessa forma poderíamos ver exatamente os resultados que queremos. LIMIT nos permite especificar exatamente quantas linhas queremos que retornem no nosso resultado.

```
SELECT primeiro_nome, SUM (venda)
FROM venda_biscoito
GROUP BY primeiro_nome
ORDER BY SUM (venda) DESC
LIMIT 2;
```

*Isto aqui diz que você quer LIMIT(limitar) o resultado para apenas os dois primeiros.*

primeiro_nome	vendas
Britney	107.91
Paris	98.23

*Esta é uma consulta longa e faz você obter apenas este pequeno resultado.*

Quando temos apenas quatro Garotas Bandeirantes na tabela e limitar este resultado para apenas dois não nos ajudar muito aqui, imagine que você estivesse trabalhando com uma tabela muito maior. Suponha que tem uma lista com as 1000 músicas que estão atualmente no topo das mais tocadas nas estações de rádios, e você quer apenas as 100 primeiras em ordem de popularidade. LIMIT permitiria que visualizasse apenas essas e não as outras 900 músicas.

**LIMIT (limite) apenas para a segunda colocada**

**LIMIT** nos permite ainda que seja apontada apenas a segunda colocada sem ter que ver a vencedora primeira colocada. Para isto, podemos utilizar **LIMIT** com dois parâmetros:

Se tentou adivinhar no que isto iria resultar, você provavelmente deve ter errado. Quando se tem dois parâmetros quer dizer algo completamente diferente que um só.

Este é o resultado para se iniciar. O Sistema SQL inicia a contagem com 0.

Esta é a quantidade de resultados para serem retornados.

primeiro_nome	venda
Britney	107.91
Paris	98.23
Nicole	96.03
Lindsay	81.08

Britney é 0,  
Paris é 1, Nicole é  
2 e Lindsay é 3

Ainda lembra das nossas 100 músicas no topo? Suponha que queremos ver da música 20 até a 30. Adicionar um parâmetro extra para nosso LIMIT realmente nos ajudaria. Nós iríamos estar simplesmente aptos a ordená-los por popularidade e adicionar ao LIMIT 19, 10. O 19 diz para se começar da 20<sup>a</sup> posição já que o Sistema conta iniciando do 0, e o 10 diz para exibir apenas 10 linhas de resultado.



## Aponte seu lápis

Escreva a consulta que nos dará a segunda-feira, e tão somente o segundo resultado utilizando a cláusula LIMIT com dois parâmetros.



## Aponte seu lápis

## Solução

Escreva a consulta que nos dará a segunda-feira, e tão somente o segundo resultado utilizando a cláusula LIMIT com dois parâmetros.

```
SELECT primeiro_nome, SUM(venda)
FROM venda_biscoito
GROUP BY primeiro_nome
ORDER BY SUM(venda) DESC
LIMIT 1;
```

Lembre-se, SQL começa contando com o 0. Então / na verdade é 2.



Meus comandos SQL estão ficando tão longos e tão complicados agora com todas estas novas palavras-chave. Eu gosto delas, elas são ótimas, mas não há um jeito de simplificar as coisas?

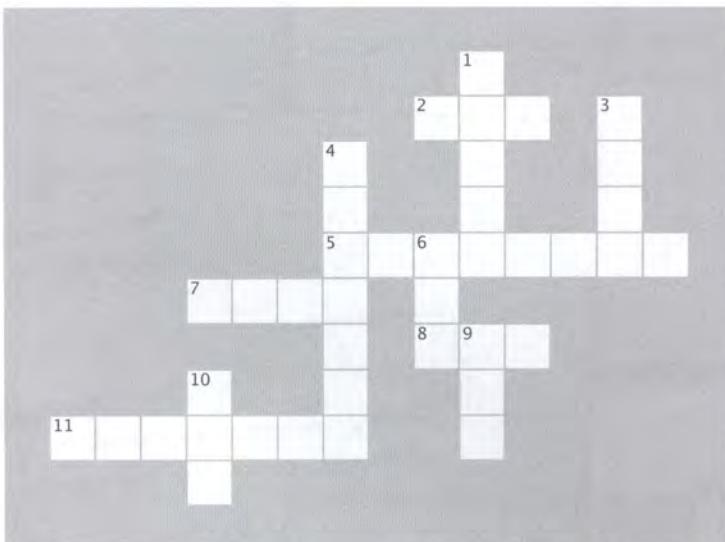
**Suas consultas estão ficando longas porque seus dados estão ficando mais complicados.**

Vamos olhar de perto a sua tabela, você deve ter aumentado. Vá para o Capítulo 7...



## Cruzadas-SELECT

É hora de dar um descanso para o lado direito do seu cérebro e deixar o lado esquerdo trabalhar: todas as palavras são relacionadas a este capítulo e ao SQL.



### Horizontais

2. Você pode encontrar o menor valor em uma coluna com esta função.
5. Esta função retorna cada valor único apenas uma vez, sem duplicidade.
7. A palavra-chave \_\_\_\_\_ no CASE permite que você diga ao seu Sistema SQL o que fazer se qualquer dos registros não alcançar os critérios.
8. O você pode encontrar o maior valor em uma coluna com esta função.
11. Utilize estas duas palavras para consolidar linhas localizadas em uma coluna em comum.

### Verticais

1. Permite que você especifique quantas linhas a serem exibidas, e em qual linha deve-se começar.
3. Se você ORDER BY (ordenar) uma coluna utilizando esta palavra-chave, o valor nove naquela coluna virá antes do 8.
4. Utilize estas duas palavras para ordenar alfabeticamente seus resultados encontrados em uma coluna que você determinar.
6. Esta função adiciona uma coluna de valores numéricos.
9. Se você ORDER BY uma coluna utilizando esta palavra-chave, o valor 8 na coluna virá antes do 9.
10. Utilize-o em um SELECT para retornar o número de resultados além dos resultados propriamente ditos.



## Sua caixa de ferramenta SQL

Você está com o Capítulo 6 nas palmas das mãos, e está de fato se deparando como todos estas funções, palavras-chave e consultas avançadas do SELECT. Para uma lista completa de dicas de ferramentas no livro, veja Apêndice iii.

### ORDER BY

Ordena alfabeticamente seus resultados baseando-se na coluna que você especifica.

### GROUP BY

Consolida linhas baseando-se em colunas comuns.

### COUNT

Pode dizer quantas linhas se encaixam em uma consulta do SELECT sem ter que visualizar as demais linhas. COUNT exibe um só valor inteiro.

### DISTINCT

Retorna cada valor único apenas uma vez, sem duplicidade.

### AVG

Exibe o valor da média de uma coluna numérica.

### MAX e MIN

Exibe o maior valor de uma coluna com o MAX, e o menor valor com o MIN.

### SUM

Soma uma coluna de valores numéricos.

### LIMIT

Permite que você especifique exatamente quantas linhas a serem exibidas e em que linha se começar.

77  
Suas novas ferramentas:  
funções, palavras-chave e  
consultas SELECT avançadas!

Respostas da página 220.

Uma porção de funções e palavras-chave com a fantasia completa estão jogando um jogo comum em festas, "Quem sou eu?" Eles darão uma pista – você tenta adivinhar quem são eles baseando-se no que eles dizem. Presuma que eles sempre falarão a verdade sobre si. Preencha as lacunas em branco à direita para identificar os participantes. Também, para cada participante, escreva se isso é uma função ou uma palavra-chave.

Participantes de hoje:

COUNT, DISTINCT, AVG, MIN, GROUP BY, SUM, MAX

## Quem sou eu?



O resultado que você obtém por me usar pode não valer muito.

**MIN** função

O que eu ponho pra fora é maior do que qualquer coisa que eu ponho pra dentro.

**SUM** função

Eu darei um resultado único.

**DISTINCT** palavra-chave

Eu direi quantos haviam ali.

**COUNT** função

Você precisa me utilizar se quiser obter uma soma.

**GROUP BY** palavra-chave

Eu estou interessado apenas nos grandes números.

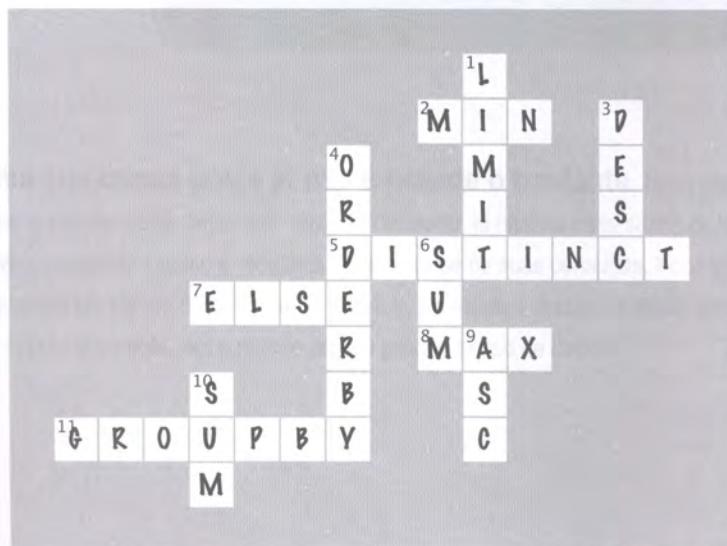
**MAX** função

Como eu sou? Algo mediano.

**AVG** função



## SELECT cross Solution





## 7 Projeto de banco de dados multitabletas

### Povoando sua tabela



**Às vezes sua tabela única já não é grande o bastante.** Seus dados se tornaram mais complexos, e aquela tabela única que você tem utilizado, já não vai mais servir. Sua tabela é cheia de dados redundantes, gastando espaço e reduzindo a velocidade de suas consultas. Você já foi o mais longe que podia com uma tabela só. Há um mundo grande lá fora e, às vezes, é preciso mais de uma tabela para conter dados, controlar e, principalmente, ser o mestre de seu próprio banco de dados.

## Encontrando uma namorada para Nigel

Nigel, o solitário amigo de Greg, tem pedido para ele ajudá-lo a encontrar uma mulher para namorar, e que tenham interesses em comum. Greg começa por puxar o registro de Nigel.

Aqui está Nigel:

```
Id_contato: 341
sobrenome: Moore
primeiro_nome: Nigel
telefone: 5552311111
email: nigelmoore@ranchersrule.com
sexo: M
aniversário: 28-08-1975
profissão: Fazendeiro
cidade: Austin
estado: TX
estado_civil: solteiro
interesses: animais, hipismo, filmes
procura: mulher solteira
```

A coluna `interesse` não é atômica; ela tem mais de um tipo de informações iguais. Ele está preocupado porque não será muito fácil de procurar.

Greg adicionou o pedido de Nigel na sua lista de AFAZERES:

Nigel



### AFAZERES

**escrever consulta para Nigel:** Irei escrever uma consulta para procurar na coluna `interesses`. Parece doloroso, terei que utilizar `LIKE`, mas será somente desta vez...

## Por que mudar alguma coisa?

Greg tinha decidido não alterar em nada esta coluna `interesses`. Ele está considerando a idéia de escrever esta consulta difícil porque não acredita que terá que escrevê-las muito freqüentemente.

Ele utiliza o campo `DATA` do aniversário para encontrar combinações de idade que não sejam mais que 5 anos mais jovem ou mais velha que Nigel.

 Aponte seu lápis

Termine a consulta personalizada para ajudar Nigel a encontrar um par compatível que compartilhe de seus interesses. Anote o que cada linha de comando faz.

```
SELECT * FROM meus_contatos
WHERE sexo = 'F'
AND estado_civil = 'solteiro'
AND estado= 'TX'
AND procura LIKE '%homem solteiro%'
AND aniversario > '1970-08-28'
AND aniversario < '1980-08-28'
AND interesses LIKE .....
AND .....
AND .....
```

 Aponte seu lápis

Solução

Termine a consulta personalizada para ajudar Nigel a encontrar um par compatível que compartilhe de seus interesses. Anote o que cada linha de comando faz.

```
SELECT * FROM meus_contatos
WHERE sexo = 'F' ← Seleciona tudo da tabela meus_contatos que combine com as seguintes condições:
AND estado_civil = 'solteiro' ← Nigel quer sair com uma mulher, então estamos procurando no sexo feminino...
AND estado= 'TX' ← ...e nós queremos que ela seja solteira.
AND procura LIKE '%homem solteiro%' ← ...e que pelo menos more no mesmo estado que Nigel.
AND aniversario > '1970-08-28' } ← Ela deveria estar procurando por um homem solteiro.
AND aniversario < '1980-08-28' } ← Ele quer alguém que não seja mais que 5 anos mais velha ou mais que 5 anos mais nova que ele.
AND interesses LIKE '%animal%' ...
AND , interesses LIKE '%cavalo%' ...
AND , interesses LIKE '%filme%' ...
```

Estes comandos vão pegar apenas as combinações de interesse de Nigel. Nós poderíamos ter usado OR aqui, mas o queremos de fato é combinar todos os seus interesses.

## A consulta funcionou muito bem

Greg encontrou o par perfeito para Nigel:

```
Id_contato: 1854
sobrenome: Fiore
primeiro_nome: Carla
telefone: 5557894855
email: cfiore@fioreanimalclinic.com
sexo: F
aniversário: 07-01-1974 ← Boa idade.
profissão: Veterinária ← Ótima profissão.
cidade: Round Rock
estado: TX ← também moro perto
estado_civil: solteira
interesses: hipismo, filmes, animais, contos de mistério, trilha
procura: homem solteiro
```

Carla e Trigger

Bonito!

Interesses compatíveis!



## Funcionou bem demais

Nigel e Carla realmente se entrosaram.

Agora Greg se tornou uma vítima de seu próprio sucesso: *todos* os seus amigos solteiros querem que ele faça uma consulta em seu banco de dados. E Greg tem muitos amigos.



Eu posso continuar escrevendo estas consultas complicadas toda noite.

Seu projeto de tabela deveria ter feito todo o trabalho pesado para você. Não escreva consultas enroladas para não se enrolar em uma tabela mal projetada.

Isto consome muito tempo. Greg adiciona uma nota em sua lista de afazeres.

### AFAZERES

**Escrever consulta para Nigel:** Eu irei escrever uma consulta para procura na coluna interesses. Parece doloroso, eu terei que utilizar **LIKE**, mas será somente desta vez...

**No futuro, ignore a coluna interesses** para consultas mais rápidas e mais fáceis.

## Resolver o problema não é a resposta

Seu amigo, Regis, pediu a Greg para encontrar um par para ele. Ele só procura de uma garota que não seja mais que cinco anos mais velha ou mais que cinco anos mais nova do que ele. Ele mora em Cambridge, MA, e ele tem interesses diferentes de Nigel.

Greg decidiu não se importar com a coluna interesses para sua consulta pequena e simples.



Regis →



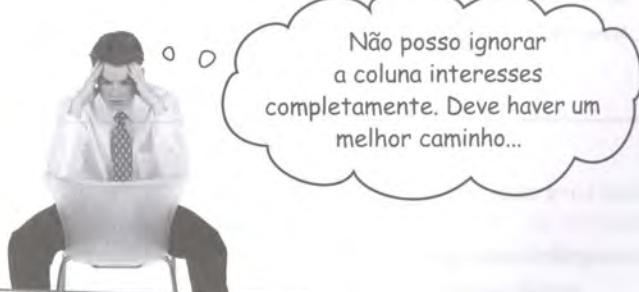
### Exercícios

Id: contato: 872  
sobrenome: Sullivan  
primeiro\_nome: Regis  
telefone: 5552311122  
email: me@kathieleeisaflake.com sexo: M  
aniversário: 20-03-1955  
profissão: Comediante  
cidade: Cambridge  
estado: MA  
estado\_civil: solteiro  
interesses: animais, trocar cartões, caça a tesouros GPS  
procura: mulher solteira

→ Resposta na página 275.

## Muitas combinações ruins

Greg entrega para Regis uma longa lista de combinações. Depois de algumas semanas, Regis liga para Greg e diz a ele que a lista é útil, e que nenhuma mulher tinha alguma coisa a ver com ele.



Não posso ignorar a coluna interesses completamente. Deve haver um melhor caminho...

### AFAZERES

**escrever consulta para Nigel:** Eu irei escrever uma consulta para procura na coluna interesses. Parece doloroso, eu terei que utilizar LIKE, mas será somente desta vez...

**No futuro, ignore a coluna interesses** para consultas mais rápidas e mais fáceis

**Consultar apenas o primeiro interesse** e ignorar o resto das informações na coluna.

Interesses SÃO importantes. Não deveríamos ignorá-los, há muitas informações valiosas ali.

## Utilizamos somente o primeiro interesse

Greg agora sabe que não podemos mais ignorar todos os interesses. Ele está assumindo que as pessoas se cadastraram colocando no interesse a ordem de importância e decide que irá consultar apenas o primeiro interesse. Suas consultas ainda são um pouco dolorosas de escrever, mas não tão ruins como quando ele incluiu LIKE para todos os interesses na coluna interesses.



Aponte seu lápis

Utilize a função SUBSTRING\_INDEX para conseguir apenas o primeiro interesse dentro da coluna interesses.



Aponte seu lápis

Solução

Utilize a função SUBSTRING\_INDEX para conseguir apenas o primeiro interesse dentro da coluna interesses.

SUBSTRING\_INDEX(interesses, ',', 1)  
Este comando pega tudo que está na frente da vírgula na coluna interesses ou linha de comando.

Aqui é '1' porque ele está procurando pela primeira vírgula. Se fosse '2', ele continuaria até achar a segunda vírgula e selecionasse tudo que o está à sua frente, que seriam os dois primeiros interesses.  
↑  
Esta é aspa pela qual o programa está procurando.

Então Greg escreve uma consulta utilizando a SUBSTRING\_INDEX e especificando que o primeiro interesse deveria se adequar com 'animais'.

```
SELECT * FROM meus_contatos
WHERE sexo = 'F'
AND estado_civil = 'solteira'
AND estado = 'MA'
AND procura LIKE '%mulher solteira %'
AND aniversario > '1950-08-28'
AND aniversario < '1960-08-28'
AND SUBSTRING_INDEX(interesses, ',', 1) = 'animais';
```

Apenas mulheres que tenha listado animais como primeiro interesse aparecerão nos resultados.

## Uma possível combinação

Finalmente! Greg encontrou um par para Regis:

contato: 459

aprenome: Ferguson

primeiro\_nome: Alexis

telefone: 5550983476

email: alexangel@yahoo.com

sexo: F

nascimento: 1956-09-19 *Boa idade.*

profissão: Artista

cidade: Pflugerville

estado: MA

status\_civil: solteiro

interesses: animais *Mora perto do Regis* *Interesses compatíveis*

## Trabalha na compatibilidade

Greg convidou Alexis para um encontro, e Greg esperou ansiosamente para ouvir como foi. Ele começou a pensar em sua tabela de contatos como um ótimo site de rede social.

No dia seguinte, Regis aparece na porta de Greg, claramente desapontado.

Regis grita, "Ela definitivamente está interessada por animais. Mas você não disse que um de seus interesses era taxidermia. Animais mortos por todo lado!"

### AFAZERES

**escrever consulta para Nigel:** Eu irei escrever uma consulta para procura na coluna interesses. Parece doloroso, eu terei que utilizar LIKE, mas será somente desta vez...

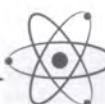
**No futuro, ignore a coluna interesses** para consultas mais rápidas e mais fáceis

**Consultar apenas o primeiro interesse** e ignorar o resto das informações na coluna.

Criar múltiplas colunas para armazenar um interesse em cada porque ter todos os interesses em uma só coluna faz a consulta ficar muito difícil.

A combinação perfeita para Regis estava na tabela, mas nunca foi descoberta porque os interesses dela estavam em ordens diferentes.

Greg decide reprojetar sua tabela.



## PODER DO CÉREBRO

Como a próxima consulta de Greg será depois que ele colocar múltiplas colunas para interesses?

## Adicionar mais colunas interesse

Greg percebe que uma só coluna de interesses torna a escrita da consulta imprecisa. Ele tem que utilizar LIKE para tentar combinar os interesses, às vezes finalizando com péssimas combinações.

Já que aprendeu como alterar tabelas recentemente, e também como separar linhas de texto, ele decide criar múltiplas colunas para interesse e colocar um interesse em cada coluna. Ele acha que quatro colunas devem ser o suficiente.



Aponte seu lápis

Use as funções ALTER e SUBSTRING\_INDEX para finalizar estas colunas. Escreva tantas consultas quanto forem necessárias.

**id\_contacto  
sobrenome  
primeiro\_nome  
telefone  
email  
sexo  
aniversario  
profissao  
cidade  
estado  
estado\_civil  
interesse1  
interesse2  
interesse3  
interesse4  
procura**

→ Respostas na página 274.

Começ

Greg está se s  
vai tentar nov

Id\_contacto  
sobrenome  
primeiro\_nome  
telefone  
email  
sexo  
aniversario  
profissao  
cidade  
estado  
estado\_civil  
interesse1  
interesse2  
interesse3  
interesse4  
procura:

## Começando de novo

Regis se sentindo péssimo por causa da experiência de Regis com Alexis, então ele tentar novamente. Ele começa puxando o registro de Regis:

```

    contato: 872
    sobrenome: Sullivan
    primeiro_nome: Regis
    telefone: 5552311122
    email: me@kathieleeisaflake.com
    sexo: M
    aniversario: 20-03-1955
    profissao: Comediante
    cidade: Cambridge
    estado: MA
    estado_civil: solteiro
    interesse1: animais
    interesse2: trocar cartões
    interesse3: caça a tesouros GPS
    interesse4: NULL
    procura: mulher solteira
  
```

} Quatro colunas de interesse na nossa recém-formatada tabela.



**Exercícios**

Então Greg escreve uma consulta personalizada para ajudar Regis a ter um encontro compatível. Ele lança em tudo que pode para ajudá-lo a encontrar uma ótima compatibilidade. Começando com as colunas mais simples – sexo, estado civil, estado, procura e aniversário – antes de iniciar a consulta de todas as colunas de interesse.

Escreva esta consulta aqui.



Então Greg escreve uma consulta personalizada para ajudar Regis a ter um encontro compatível. Ele lança em tudo que pode para ajudá-lo a encontrar uma ótima compatibilidade. Começando com as colunas mais simples – sexo, estado civil, estado, procura e aniversário – antes de iniciar a consulta de todas as colunas de interesse.

Escreva esta consulta aqui.

## Tudo

Adiciona não ajuda resolver o projeto deixa co seja fácil tabela vi dados at

```
SELECT * FROM meus_contatos
```

```
WHERE sexo = 'F'  
AND estado_civil = 'solteira'  
AND estado = 'MA'  
AND procura LIKE '%homem solteiro%'  
AND aniversario > '1950-03-20'  
AND aniversario < '1960-03-20'  
AND
```

```
(
```

```
interesse1 = 'animais'  
OR interesse2 = 'animais'  
OR interesse3 = 'animais'  
OR interesse4 = 'animais'  
)  
AND
```

```
(
```

```
interesse1 = 'trocar cartões'  
OR interesse2 = 'trocar cartões'  
OR interesse3 = 'trocar cartões'  
OR interesse4 = 'trocar cartões'  
)  
AND
```

```
(
```

```
interesse1 = 'caça a tesouros GPS'  
OR interesse2 = 'caça a tesouros GPS'  
OR interesse3 = 'caça a tesouros GPS'  
OR interesse4 = 'caça a tesouros GPS'  
)
```

} Regis quer namorar uma garota solteira e nascida entre 1970 e 1980, que mora em Massachusetts que gosta de namorar um homem solteiro.

Greg tem que olhar entre cada coluna de interesse para verificar se os valores não combinam com os interesses de Regis, já que poderia haver uma combinação em cada uma destas quatro novas colunas.

Regis tinha uma coluna de interesse NULL; portanto, nós temos que procurar somente por três interesses e não quatro.

## Tudo foi em vão...

Adicionar as novas colunas não ajudou em nada a resolver o problema básico; o projeto da tabela não deixa com que a consulta seja fácil. Cada versão da tabela viola as regras dos dados atômicos.

Esta pareceu ser uma solução tão boa, mas ela fez a consulta ficar ainda mais complicada.

### AFAZERES

**escrever consulta para Nigel:** Eu irei escrever uma consulta para procura na coluna interesses. Parece doloroso, eu terei que utilizar **LIKE**, mas será somente desta vez...

**No futuro, ignore a coluna interesses** para consultas mais rápidas e mais fáceis

**Consultar apenas o primeiro interesse** e ignorar o resto das informações na coluna.

Criar múltiplas colunas para armazenar um interesse em cada porque ter todos os interesses em uma só coluna faz a consulta ficar muito difícil.

?

Mas espere



Nós poderíamos criar uma  
tabela que só tivesse interesses?  
Isto ajudaria?



### PODER DO CÉREBRO

Adicionar uma nova tabela ajudaria? Como poderíamos conectar os dados de uma nova tabela com uma tabela já existente?

## Pense além da tabela única

Sabemos que não há uma boa solução se trabalharmos dentro da nossa tabela existente. Nós tentamos diversas maneiras de consertar os dados, até alterando a estrutura da tabela única. Nada funcionou.

Precisamos pensar além desta tabela. O que realmente precisamos são de **mais tabelas** que possam **trabalhar com** a existente permitindo-nos **associar cada pessoa com mais de um interesse**. E isso nos permitirá manter os dados existentes de forma intacta.

**Nós precisamos mover nossas colunas não-atômicas da nossa tabela para novas tabelas.**

```
File Edit Window Help MessyTable
>DESCRIBE meus_contatos;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| id_contato | int(11)    | NO   | PRI | NULL    | auto_increment |
| sobrenome  | varchar(30) | YES  |     | NULL    |                |
| primeiro_nome | varchar(20) | YES  |     | NULL    |                |
| telefone   | varchar(10)  | YES  |     | NULL    |                |
| email      | varchar(50)  | YES  |     | NULL    |                |
| sexo        | char(1)     | YES  |     | NULL    |                |
| aniversario | date       | YES  |     | NULL    |                |
| profissao  | varchar(50)  | YES  |     | NULL    |                |
| cidade      | varchar(50)  | YES  |     | NULL    |                |
| estado      | varchar(2)   | YES  |     | NULL    |                |
| estado_civil | varchar(20)  | YES  |     | NULL    |                |
| interesses  | varchar(100) | YES  |     | NULL    |                |
| procura     | varchar(100) | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+
13 rows in set (0.01 sec) >
```

O ban

Ainda lem  
O proble  
nós altera

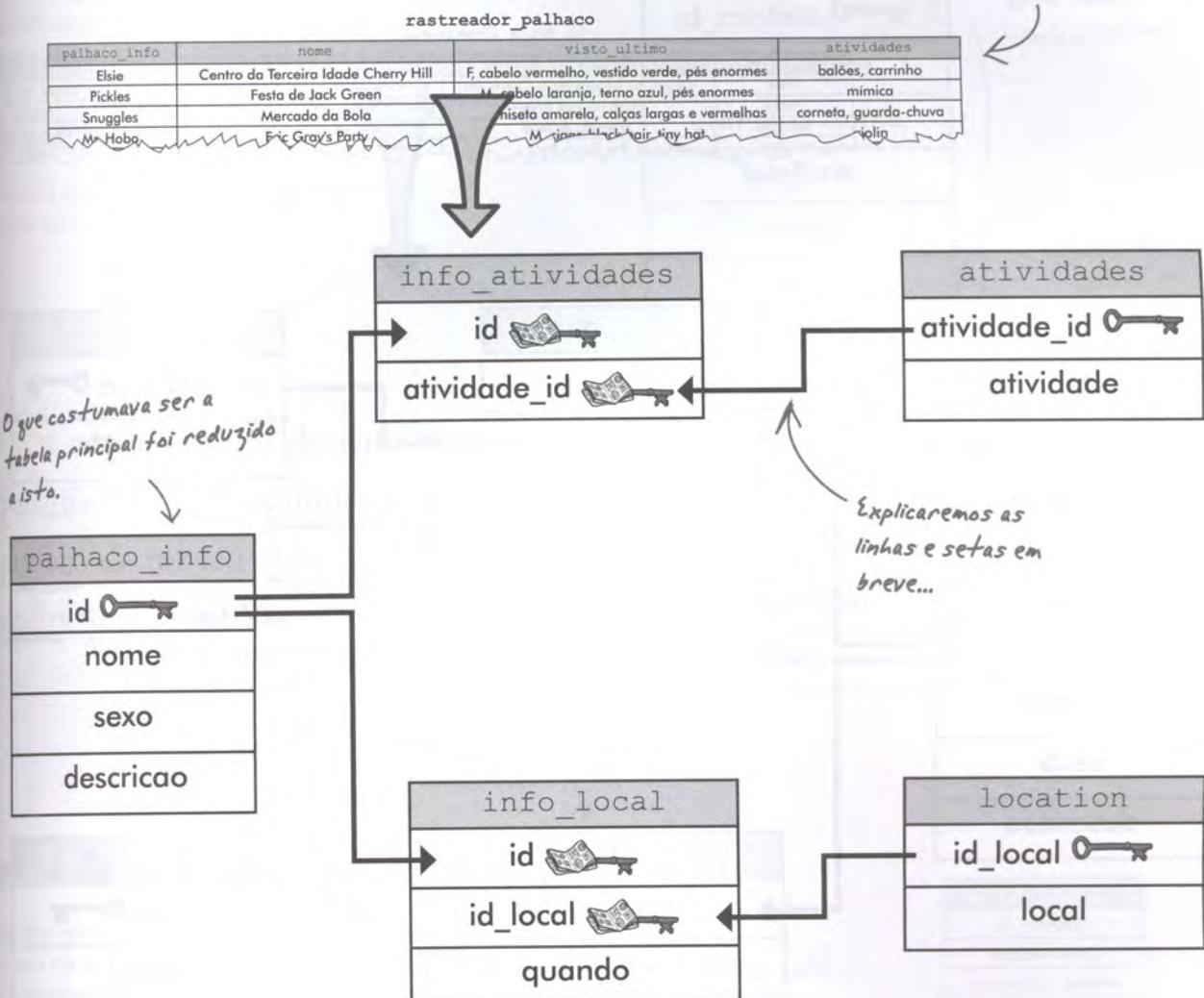
O  
+  
a

Nas pró  
tabela fo  
e chaves  
superad  
estas reg

## Banco de dados rastreador de palhaços de múltiplas tabelas

Toda lembraria da nossa tabela rastreadora de palhaços do capítulo 3? O problema dos palhaços em Dataville continua aumentando, então alteramos a tabela única em um conjunto de tabelas muito mais útil.

Como a tabela rastreador-palhaco costumava ser.



Nas próximas páginas você verá porque a tabela foi dividida desta forma e o que as setas e chaves querem dizer. Quando tivermos superado tudo isso, poderemos aplicar todas estas regras para a gregs\_list.

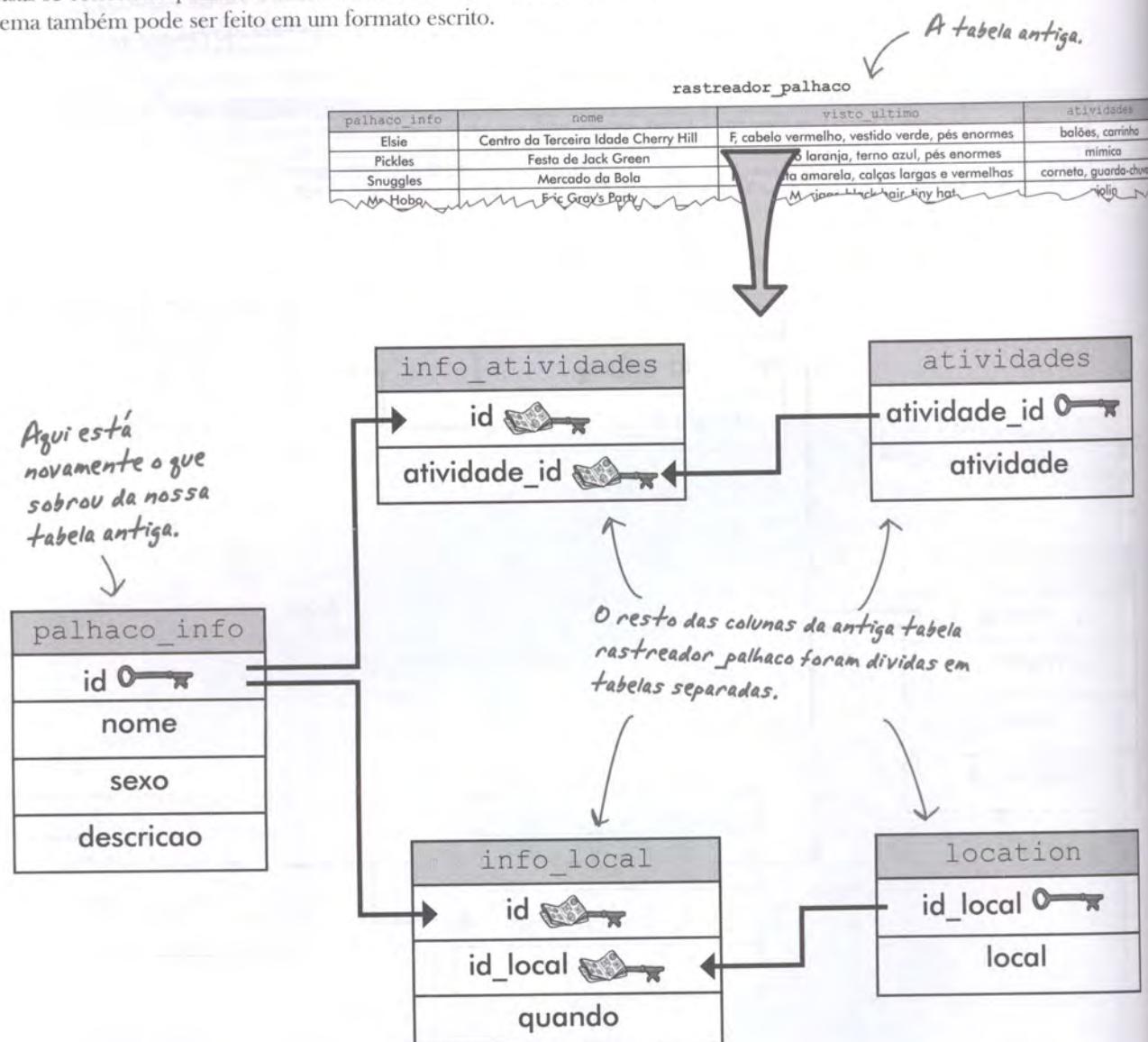


O que você acha que as linhas com setas significam? E quanto aos símbolos de chaves?

# O esquema do banco de dados rastreador\_palhaco

Uma representação de todas as estruturas, tanto as tabelas como as colunas, bem como a forma como elas se conectam, é conhecida como **esquema**.

Criar uma descrição visual de seu banco de dados pode ajudá-lo a ver como as coisas se conectam quando você está escrevendo suas consultas, mas seu esquema também pode ser feito em um formato escrito.



Uma descrição dos dados (as colunas e as tabelas) no seu banco de dados, junto com qualquer outro objeto relacionado e a forma como todos eles se conectam, é conhecida como **ESQUEMA**.

## Uma forma mais fácil de diagramar suas tabelas

Já viu no que a tabela rastreador de palhaços convertida. Vamos ver como podemos consertar tabela `meus_contatos` da mesma forma.

Este ponto, todas as vezes que olhamos para tabela, nós a visualizamos através dos nomes das colunas ao longo da parte superior dos dados, então utilizamos um comando DESCRIBE em uma janela de terminal. Estas duas formas são usadas para tabelas únicas, mas não são muito úteis para serem utilizadas quando queremos um diagrama de múltiplas tabelas.

Aqui está uma técnica abreviada para diagramar a tabela `meus_contatos`:

Criar um diagrama de sua tabela permite que você mantenha o projeto da tabela separado dos dados que estão dentro dela.

meus_contatos
<code>id_contato</code>
<code>sobrenome</code>
<code>primeiro_nome</code>
<code>telefone</code>
<code>email</code>
<code>sexo</code>
<code>aniversario</code>
<code>profissao</code>
<code>cidade</code>
<code>estado</code>
<code>estado_civil</code>
<code>interesses</code>
<code>procura</code>

## Como ir de uma tabela para duas

Nos sabemos que a coluna `interesses` é realmente difícil de se consultar na forma como ela se encontra agora. Ela tem valores múltiplos na mesma coluna. E mesmo quando tentamos criar múltiplas colunas para ela, nossas colunas ficaram bastante difíceis de serem escritas.

Aqui está nossa atual tabela `meus_contatos`. Nossa coluna `interesses` não é atômica, e há somente uma boa maneira de fazê-la atômica: precisamos de uma nova tabela que irá armazenar todos os interesses.

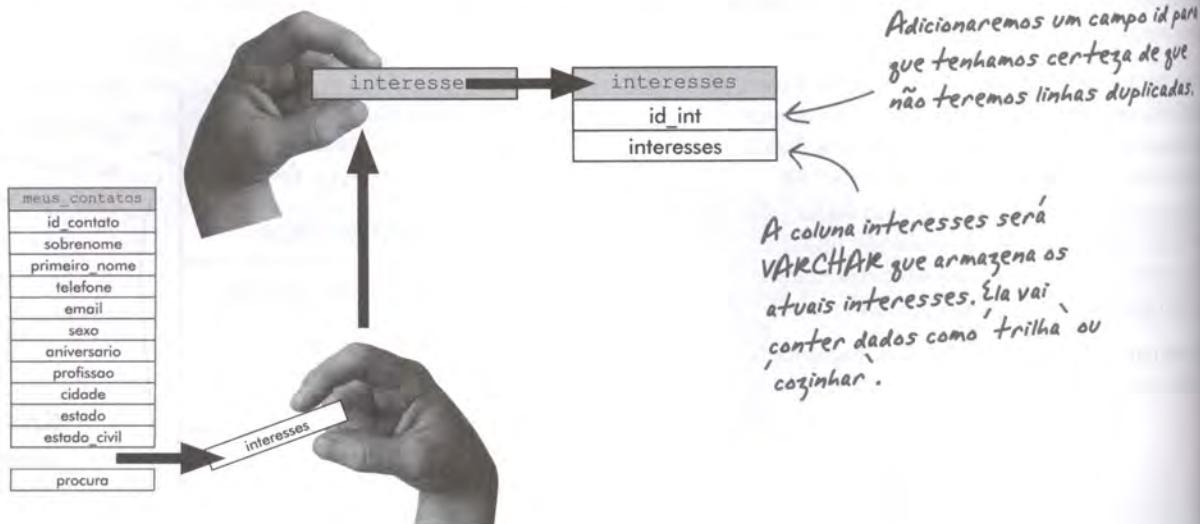
Vamos começar desenhando alguns diagramas de como nosso tabela poderia ser. Não vamos de fato criar uma nova tabela ou tocar em qualquer dos dados até concluirmos nosso novo esquema.

meus_contatos
<code>id_contato</code>
<code>sobrenome</code>
<code>primeiro_nome</code>
<code>telefone</code>
<code>email</code>
<code>sexo</code>
<code>aniversario</code>
<code>profissao</code>
<code>cidade</code>
<code>estado</code>
<code>estado_civil</code>
<code>interesses</code>
<code>procura</code>

1

**Remova a coluna interesses e coloque-a em sua própria tabela.**

Aqui nós movemos a coluna interesses para uma nova tabela.



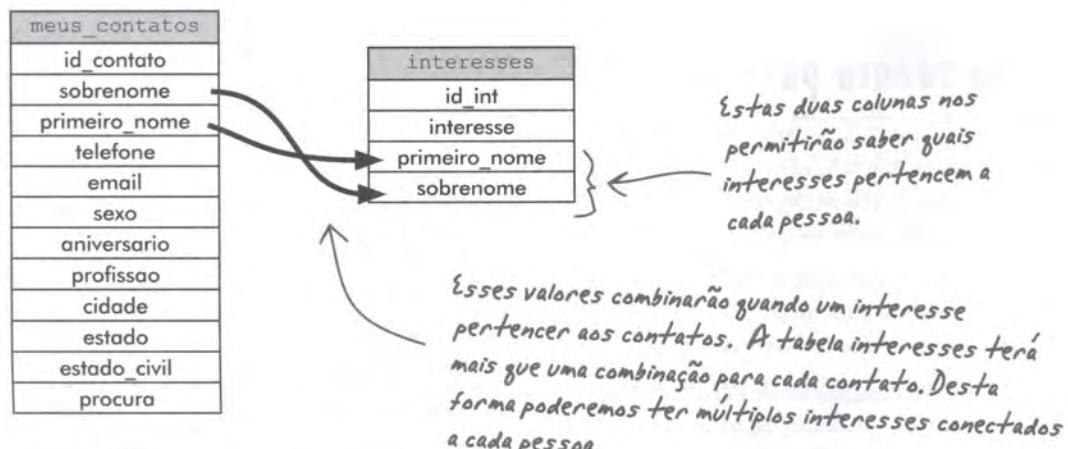
Nossa nova tabela interesses todos os interesses da tabela `meus_contatos`, um interesse por linha.

2

**Adicione colunas que nos permitirá identificar quais interesses pertencem para cada pessoa na tabela `meus_contatos`.**

Nós movemos nossa coluna interesses para fora da `meus_contatos`, mas não temos nenhum jeito de saber quais interesses pertencem para cada pessoa. Precisamos usar informações da tabela `meus_contatos` e colocá-las na tabela interesses para criar uma conexão entre elas.

Uma maneira possível é adicionar as colunas `primeiro_nome` e `sobrenome` na tabela interesses.



Nós temos a idéia certa, mas `primeiro_nome` e `sobrenome` não são as melhores opções de colunas para conectar estas duas tabelas.

## Conectando suas tabelas em um diagrama

Olhar de perto para nossa idéia para a tabela meus\_contatos.

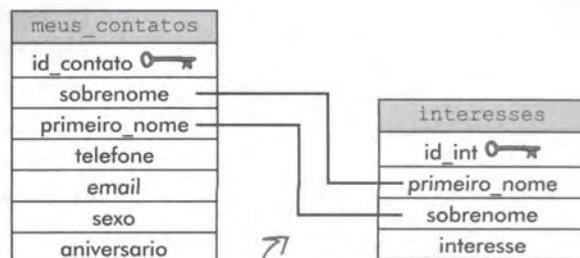
para  
que  
das.

É aqui está nosso rascunho inicial:



De alguma forma conecte através de  
primeiro\_nome e sobrenome, que estão  
nas duas tabelas nos dizendo quem  
tem qual interesse.

É aqui está nosso esquema:



Estas linhas mostram como os dados  
se conectam. Você poderia desenhá-  
las sem os ângulos, mas eles fazem  
as linhas se tornarem mais fácil de  
seguir-las.

Veja como as linhas com ângulos retos se curvam para as colunas com as quais se combinam  
na outra tabela onde o esquema nos permite arrumar nosso rascunho de uma forma que qualquer  
programador SQL entenderá, já que são utilizados símbolos padrões.

É aqui está uma série de comandos SELECT que nos permitirá utilizar os dados em ambas as tabelas.

1  
SELECT primeiro\_nome, sobrenome  
FROM meus\_contatos  
WHERE (uma porção de condições)

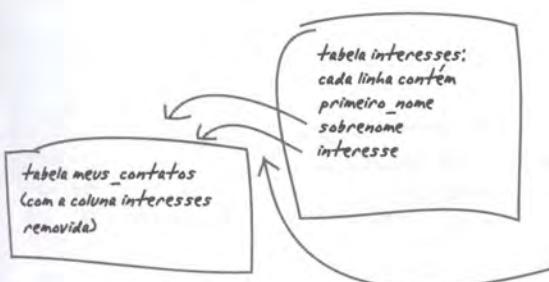
2  
SELECT interesse FROM interesses  
WHERE primeiro\_nome = 'algum nome'  
AND sobrenome = 'sobrenome';

Não se preocupe se elas parecem ineficientes. É apenas para  
lhe mostrar como os dados de uma tabela podem ser utilizados  
para puxar dados de outra. (Vamos lhe mostrar uma forma  
melhor em breve)

Aponte seu lápis

Utilize este espaço para rascunhar mais idéias para adicionar novas tabelas para  
o banco de dados gregs\_list para nos ajudar a rastrear múltiplos interesses.

Não se preocupe em fazer um esquema caprichado; estamos no estágio das  
idéias agora. Uma idéia já foi bolada para você, mas tem uma falha.



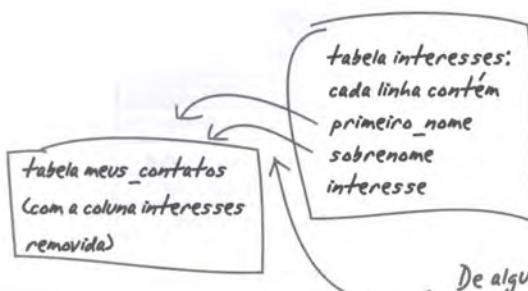
De alguma forma conecte  
através de primeiro\_nome e  
sobrenome que estão nas duas  
tabelas nos dizendo quem tem  
qual interesse.



## Aponte seu lápis Solução

Utilize este espaço para rascunhar mais idéias para adicionar novas tabelas para o banco de dados gregs\_list para nos ajudar a rastrear múltiplos interesses.

Não se preocupe em fazer um esquema caprichado; estamos no estágio das idéias agora. Uma idéia já foi bolada para você, mas tem uma falha.



Utilizando o primeiro\_nome e sobrenome para se conectar à tabela interesses não é uma boa idéia. Mais de uma pessoa na tabela meus\_contatos pode ter o mesmo nome e sobrenome, então poderíamos estar conectando pessoas a interesses equivocados. Estaremos em melhor posição se utilizarmos nossa chave primária para fazer a conexão.

De alguma forma conecte através de  
primeiro\_nome e sobrenome que estão  
nas duas tabelas nos dizendo quem  
tem qual interesse.

Ao invés de utilizar primeiro\_nome e sobrenome que podem não ser verdadeiramente únicos, nós poderíamos utilizar a coluna id\_contato para conectar ambas as tabelas:



Ao utilizar id\_contato, ficamos com um valor verdadeiramente único. Nós sabemos que os interesses com um id\_contato em particular pertencem absolutamente àquela linha correspondente na tabela meus\_contatos.

## Conectando suas tabelas

O problema com o nosso primeiro rascunho de tabelas conectadas é que nós estávamos tentando utilizar os campos primeiro\_nome e sobrenome para, de alguma forma, conectar as duas tabelas. Mas se duas pessoas na tabela meus\_contatos tiverem o mesmo primeiro\_nome e sobrenome?



Se duas pessoas tiverem o mesmo nome e sobrenome, podemos ficar com seus interesses misturados.

precisamos de uma coluna **única** para conectá-las. Ainda bem que já começamos a normalizar a base de dados e temos de fato uma coluna única em `meus_contatos`: a **chave primária**.

Podemos utilizar os valores da chave primária da tabela `meus_contatos` como uma coluna na tabela `interesses`. Melhor ainda, saberemos quais interesses pertencem a cada pessoa na tabela `meus_contatos` através desta coluna. É chamada de **chave estrangeira**.

meus_contatos
<code>id_contato</code> 0
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
profissao
cidade
estado
estado_civil
procura

Estamos nos certificando que esta nova tabela está na Primeira Forma Normal ao dar a cada registro sua própria chave primária.

interesses
<code>id_int</code> 0
interesses
<code>id_contato</code>

A CHAVE ESTRANGEIRA nos diz a que interesses pertencem para cada pessoa na tabela `meus_contatos`.



**A FOREIGN KEY** (**chave estrangeira**) é uma coluna em uma tabela que faz referência à **PRIMARY KEY** (**chave primária**) de outra tabela.

## Atos sobre a CHAVE EXTERNA



Uma chave estrangeira pode ter um nome diferente do que o nome da chave primária de onde se originou.

A chave primária usada por uma chave estrangeira é também conhecida como *Parent Key*. A tabela onde a chave primária se encontra é chamada de *Parent Table*.

A chave estrangeira pode ser usada para se ter certeza de que as linhas de uma tabela têm linhas correspondentes em outra tabela.

Os valores da chave estrangeira pode ser null, ainda que os valores da chave primária não possam.

Chave estrangeira não precisa ser única – na verdade, geralmente não são.



Eu entendi que uma chave estrangeira permite que eu conecte duas tabelas. Mas para que serve uma chave estrangeira NULL? Há alguma maneira de se ter certeza que sua chave estrangeira está conectada a uma parent key?

**Uma chave estrangeira NULL significa que não há uma chave primária correspondente na parent table.**

Mas podemos ter certeza de que uma chave estrangeira contém um valor significativo, que exista uma parent table, utilizando uma **constraint**.

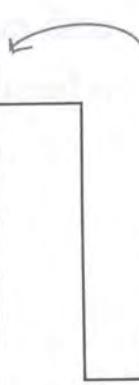
## Restringindo sua chave estrangeira

Embora pudesse simplesmente criar uma tabela e colocar uma coluna agindo como chave estrangeira, não se é uma chave estrangeira de verdade a não ser que você faça essa atribuição no momento que utilizar o CREATE ou ALTER TABLE. A chave é criada dentro de uma estrutura chamada **constraint (restrição)**.

Pense em uma **CONSTRAINT (RESTRIÇÃO)** como  
uma regra que nossa tabela tem que seguir.

**Você estará apto a inserir valores em uma chave estrangeira que existe na tabela de onde aquela chave veio, a parent table. Isto é chamado de Integridade Referencial.**

meus_contatos
id_contato
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
profissao
cidade
estado
estado_civil
procura



Nossa tabela original **meus\_contatos** é uma **parent table** desde que parte de seus dados foi movida para uma nova tabela chamada...

tabela infantil

interesses
id_int
interesses
id_contato

← Integridade Referencial quer dizer que você só pode colocar valores na chave externa que já existem na parent table.

**Você só pode utilizar uma chave estrangeira para fazer referência a um valor único na parent table.**

**Não é necessário que seja a chave estrangeira seja a chave primária de uma tabela, só basta ser um valor único.**

## Por que se importar com chaves estrangeiras?



Ok, eu sei que ao puxar os interesses para fora da tabela **meus\_contatos** é único jeito para eu estar apto a fazer consultas mais facilmente. E Regis realmente precisa encontrar alguém legal... Agora o que realmente preciso é saber **COMO** criar uma tabela com uma chave estrangeira.

**Você pode adicionar uma chave estrangeira quando criar uma nova tabela.**

E você pode inserir uma chave estrangeira com **ALTER TABLE**. A sintaxe é simples. Você precisa saber o nome da chave primária na parent table bem como o nome da parent table. Vamos criar a tabela **interesses** com uma chave estrangeira, **id\_contato** da tabela **meus\_contatos**.

**Criar uma CHAVE ESTRANGEIRA** como uma restrição tem vantagens bem definidas.

Você terá erros se violar as regras, que não permitirá que faça algo acidentalmente que vá danificar a tabela.

## Perguntas Idiotas

**P:** Depois de termos tirado os interesses de meus\_contatos, como vou consultá-los?

**R:** Faremos isso no próximo capítulo. Você verá que é realmente fácil escrever consultas capazes de puxar dados de múltiplas tabelas. Mas, por ora, precisamos projetar novamente meus\_contatos para fazer nossas consultas serem simples e eficientes.

## criar uma tabela com uma CHAVE ESTRANGEIRA

Sabemos que você sabe porque deveria criar uma chave estrangeira com uma constraint, aqui está como vai poder fazer isso. Note como estamos nomeando a constraint para podermos dizer de qual tabela a chave veio.

```
CREATE TABLE interesses (
```

Id\_int INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

Adicionar o comando CHAVE PRIMÁRIA para a linha onde você a definiu é outra maneira (mais rápida) de designar sua chave primária.

interesse VARCHAR(50) NOT NULL, um jeito que nos diga de qual tabela veio esta chave (meus\_contatos), como nomeamos a chave (id\_contato) e que ela é uma chave estrangeira (fk).

id\_contato INT NOT NULL,

CONSTRAINT meus\_contatos\_id\_contato\_fk

FOREIGN KEY (id\_contato)

REFERENCES meus\_contatos (id\_contato);

Estamos nomeando esta CONSTRAINT de interesse, um jeito que nos diga de qual tabela veio esta chave (meus\_contatos), como nomeamos a chave (id\_contato) e que ela é uma chave estrangeira (fk).

Se mudarmos de ideia posteriormente, este é o nome que costumamos desfazer. Esta linha é opcional, mas é uma boa prática usá-la.

O nome da coluna em parênteses é o que será a chave estrangeira. Você pode colocar o nome que quiser.

Exercícios



Experimente você mesmo. Abra seu console e digite o código acima para criar sua própria tabela interesses.

Quando tiver criado, dê uma olhada na estrutura de sua nova tabela. Que nova informação você vê que diz que há uma constraint ali?



## Solução dos Exercícios

Experimente você mesmo. Abra seu console e digite o código acima para criar sua própria tabela interesses.

Quando tiver criado, dê uma olhada na estrutura de sua nova tabela. Que nova informação você vê que diz que há uma constraint ali?

```
File Edit Window Help
> DESC interesses;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| int_id | int(11) | NO | PRI | NULL | auto_increment |
| interesse | varchar(50) | NO | | | |
| id_contato | int(11) | NO | MUL | | |
+-----+-----+-----+-----+-----+
```

*MUL quer dizer que Múltiplas ocorrências de um mesmo valor poderão ser armazenadas nesta coluna. Isto é, nos permite rastrear os diversos interesses para cada id\_contato na meus\_contatos.*

---

## não existem Perguntas Idiotas

---

**P:** Você tem este trabalho para criar uma constraint para a chave externa, mas por quê? Você não poderia simplesmente usar a chave de outra tabela e chamá-la de chave externa sem adicionar a constraint?

**R:** Poderia. Mas criá-la como uma constraint você seria capaz de inserir valores nela que existem na parent table. Isto fortalece o vínculo entre as duas tabelas.

**P:** "Fortalece o vínculo"? O que isto quer dizer?

**R:** A constraint da chave externa assegura a integridade referencial (em outras palavras, certifica-se de que se houvesse uma linha em uma tabela com uma chave externa, ela deve corresponder com uma linha em outra tabela através da chave primária). Se tentar deletar a linha em uma tabela com a chave primária ou alterar o valor de uma chave primária, você vai obter um erro se o valor da chave primária é uma chave externa restrita em outra tabela.

**P:** Isto quer dizer então que nunca poderei deletar uma linha do meus\_contatos que tenha uma chave primária se ela constar na tabela interesses como uma chave externa?

**R:** Você pode, apenas deve remover a chave externa, primeiro. Afinal se está removendo alguém da meus\_contatos, não há necessidade de saber quais são seus interesses.

**P:** Mas quem se importa se eu deixar aquelas linhas sobressalentes na tabela interesses?

**R:** Isto é demorado. Aquelas linhas são chamadas de órfãs, e elas podem fazer diferença em seu tempo de espera. Tudo que elas vão fazer é deixar as consultas mais lentas causando uma procura em dados inúteis.

**P:** Ok, estou convencido. Há outras constraints (restrições) além da chave externa?

**R:** Você já viu a constraint da chave primária. Utilizando a palavra-chave **UNIQUE** (ao criar a coluna), isto é, considerado uma constraint. Há também um tipo não disponível no MySQL, chamada uma **CHECK** constraint. Ela permite que você especifique uma condição que deve ser atendida em uma coluna antes que insira um valor nela. Você deverá consultar a documentação específica de seu Sistema SQL para mais informações sobre a **CHECK**.

## Relacionamentos entre tabelas

... podemos como conectar as tabelas através de chaves externas, mas ainda precisamos considerar como as tabelas se relacionam umas com as outras. Na tabela `meus_contatos`, nosso problema é que precisamos associar **muitas pessoas** com **muitos interesses**.

É um dos três padrões de possibilidades que você verá repetitivamente com seus dados: **um-a-um**, **um-para-muitos** e **muitos-para-muitos**, e uma vez que identificar o relacionamento que seus dados se ajustam, concluindo com o design de tabelas múltiplas - seu esquema - torna-se muito simples.

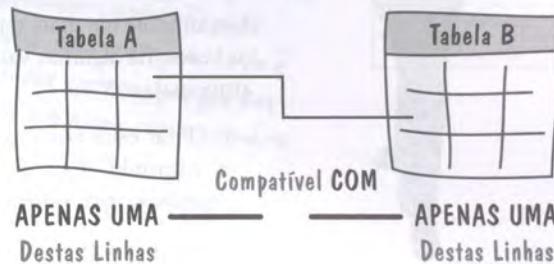
### Padrões de dados: um-a-um

Vamos observar o primeiro padrão, **um-a-um**, e ver como é aplicado. Neste padrão, um registro na Tabela A deve ter EXATAMENTE um registro compatível na Tabela B.

Por exemplo, digamos que a Tabela A contenha seu nome e a Tabela B contenha os detalhes de seu salário e CPF, com a intenção de isolar estes dois dados do resto da tabela para mantê-los de forma mais segura.

**Tanto as tabelas conterão seu número ID para que você obtenha o salário corretamente.** O funcionário na parent table é uma chave primária, o funcionário na child\_table é uma chave externa.

No esquema, as linhas conectadoras são **somente** para mostrar que estamos vinculando **uma** coisa **com uma** coisa.



Cada pessoa na tabela funcionários pode ter somente um número do CPF, e cada CPF indica apenas uma pessoa. Uma pessoa e um CPF fazem deste relacionamento do tipo um-a-um.

funcionários			salário		
<code>id_funcionario</code>	<code>primeiro_nome</code>	<code>sobrenome</code>	<code>CPF</code>	<code>salario_nivel</code>	<code>id_funcionario</code>
1	Beyonce	Knowles	234567891	2	6
2	Shawn	Carter	345678912	5	35
3	Shakira	Ripoll	123456789	7	1

Estas tabelas também têm um relacionamento um-a-um, já que a chave primária da tabela `funcionário`, `id_funcionario`, está sendo usada como chave externa da tabela `salário`.

## Padrões de dados: quando usar tabelas um-a-um



Então devemos colocar todos os nossos dados um-a-um em novas tabelas?

**Na verdade não. Não vamos usar este tipo de relacionamento tão freqüentemente.**

Há apenas algumas razões para que você conecte suas tabelas em relacionamentos um-a-um.

### Quando usar tabelas um-a-um

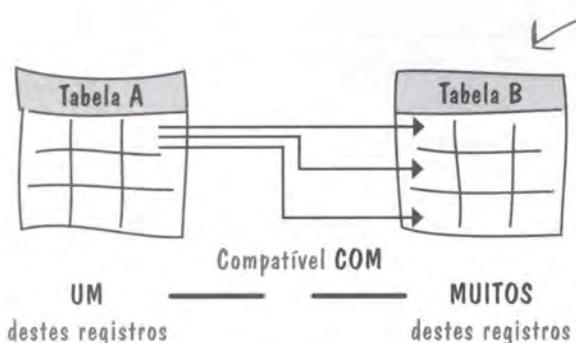
Geralmente faz mais sentido deixar seus dados um-a-um na sua tabela principal, mas, às vezes, há algumas vantagens que você tem ao puxar estas colunas para fora da principal.

1. Tirar estas colunas permite que você escreva consultas mais rápidas. Por exemplo, se na maioria das vezes precisar consultar o CPF e não muito outras coisas, você poderia consultar apenas a tabela pequena.
2. Se você tem uma coluna contendo valores que ainda não conhece, pode isolá-la e evitar valores NULL na sua tabela principal.
3. Você pode querer tornar alguns dados mais acessíveis. Isolando-os, permite um acesso restrito até eles. Por exemplo, você tem uma tabela de funcionários e pode querer guardar a informação acerca do salário fora da tabela principal.
4. Se você tem um grande pedaço de dados, uma do tipo BLOB, por exemplo, pode querer guardar os dados mais pesados em uma tabela separada.

**Um-a-um:** exatamente uma linha da parent table é relacionada a uma linha da child table.

## Padrões de dados: um-para-muitos

Um-para-muitos quer dizer que em um registro na Tabela A podem ter vários registros compatíveis na Tabela B, mas os registros da Tabela B podem ter apenas **um registro** na Tabela A.



Um registro na Tabela A é compatível com MUITOS registros na Tabela B, mas qualquer registro da Tabela B pode ser compatível com apenas UM registro na Tabela A.

**Um-para-muitos:** um registro na Tabela A pode ter MUITOS registros compatíveis na Tabela B, mas um registro na Tabela B pode ter apenas UM registro na Tabela A.

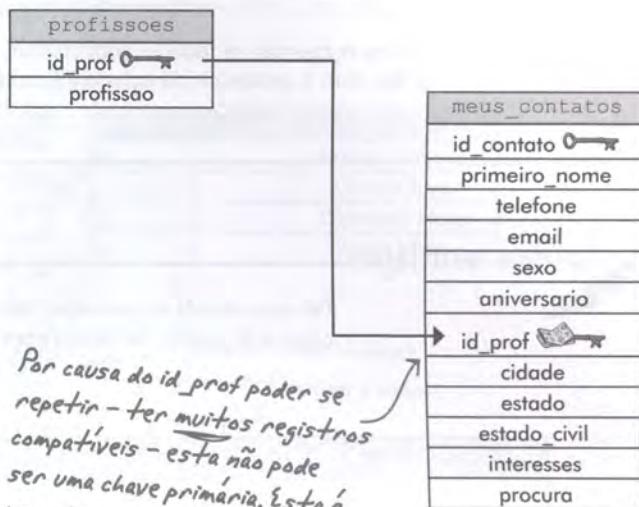
*id\_prof* na tabela *meus\_contatos* é um bom exemplo de um relacionamento um-para-muitos. Cada pessoa tem um *id\_prof*, mas mais do que uma pessoa em *meus\_contatos* podem ter o mesmo *id\_prof*.

No exemplo, nós movemos a coluna *profissão* para uma child table e alteramos a coluna *profissão* na parent para uma chave externa, a coluna *id\_prof*. Já que este é um relacionamento um-para-muitos, podemos utilizar a *id\_prof* para unir as tabelas para nos permitir conectá-las.

A conexão tem uma **seta preta** no final para mostrar que estamos vinculando **uma** coisa com **muitas coisas**.

Cada linha na tabela *profissões* pode ter muitas linhas compatíveis em *meus\_contatos*, mas cada linha em *meus\_contatos* é apenas uma linha compatível na tabela *profissões*.

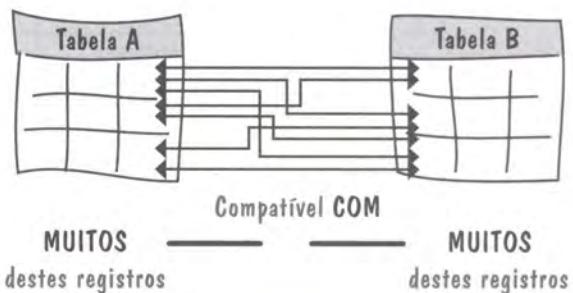
No exemplo, a coluna *id\_prof* para programador pode aparecer mais de uma vez na tabela *meus\_contatos*, mas cada linha em *meus\_contatos* terá apenas um *id\_prof*.



## Mulheres de dados: chegando ao muitos-para-muitos

Muitas mulheres possuem **muitos** pares de sapato. Se criássemos uma tabela contendo mulheres e outra tabela contendo sapatos a serem todos vinculados, precisaríamos vincular muitos registros a muitos registros já que cada uma mulher pode possuir um mesmo modelo de sapato.

Suponha que Carrie e Miranda compraram chinelo da Old Navy e botas Prada, e Samantha e Miranda têm ambas a mesma Sandália de laço Manolo, e Charlotte tem um de cada. Aqui está como o vínculo entre as mulheres e os sapatos seria.

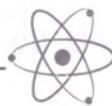


<i>id_mulher</i>	<i>mulher</i>	<i>id_sapato</i>	<i>nome_sapato</i>
1	Carrie	1	Sandália de Laço Manolo
2	Samantha	2	Tamancos de Crocodilo
3	Charlotte	3	Chinelos Old Navy
4	Miranda	4	Botas Prada

Imagine que elas amaram tanto seus sapatos que todas compraram um par daqueles que ainda não tinham. Aqui está como ficou o mosaico de cada mulher em relação aos pares de sapatos.

*As linhas conectadoras têm setas pretas em ambas as extremidades; nós estamos vinculando muitas coisas a muitas coisas.*

<i>id_mulher</i>	<i>mulher</i>	<i>id_sapato</i>	<i>nome_sapato</i>
1	Carrie	1	Sandália de Laço Manolo
2	Samantha	2	Tamancos de Crocodilo
3	Charlotte	3	Chinelos Old Navy
4	Miranda	4	Botas Prada



## PODER DO CÉREBRO

Como podemos consertar as tabelas sem colocar mais de um valor em uma coluna (e ficar enrolando como Greg fez com o problema da coluna interesses nas suas consultas para Regis)?



Aponte seu lápis

Dê uma olhada no primeiro par de tabelas. Tentamos consertar o problema adicionando a coluna id\_sapato na tabela com os registros das mulheres como uma chave externa.

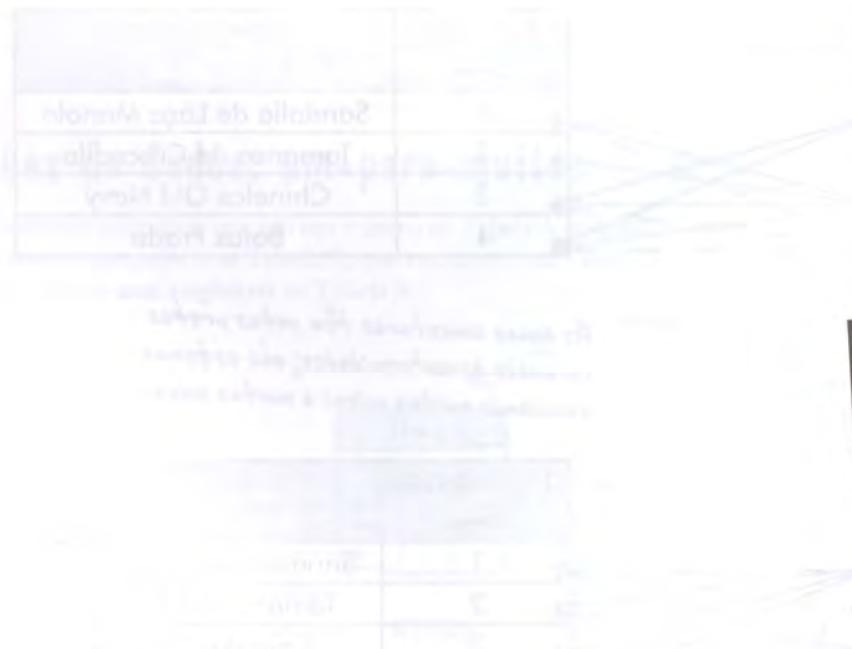
<b>id_mulher</b>	<b>mulher</b>	<b>id_sapato</b>
1	Carrie	3
2	Samantha	1
3	Charlotte	1
4	Miranda	1
5	Carrie	4
6	Charlotte	2
7	Charlotte	3
8	Charlotte	4
9	Miranda	3
10	Miranda	4

<b>id_sapato</b>	<b>nome_sapato</b>
1	Manolo Strappies
2	Crocs Clogs
3	Old Navy Flops
4	Prada boots

Agora as duas tabelas se conectam com a coluna id\_sapato.

Faça o rascunho das tabelas você mesmo, só que desta vez coloque a coluna id\_mulher na tabela de sapatos como uma chave externa.

Quando estiver acabado, desenhe os vínculos.



## Aponte seu lápis

**Solução** Dê uma olhada no primeiro par de tabelas. Tentamos consertar o problema adicionando a coluna id\_sapato na tabela com os registros das mulheres como uma chave externa.

id_mulher	mulher	id_sapato
1	Carrie	3
2	Samantha	1
3	Charlotte	1
4	Miranda	1
5	Carrie	4
6	Charlotte	2
7	Charlotte	3
8	Charlotte	4
9	Miranda	3
10	Miranda	4

id_sapato	nome_sapato
1	Manolo Strappies
2	Crocs Clogs
3	Old Navy Flops
4	Prada boots

Agora as duas tabelas se conectam com a coluna id\_sapato.

Perceba os dados duplicados nas colunas mulher e nome\_sapato.

Faça o rascunho das tabelas você mesmo, só que desta vez coloque a coluna id\_mulher na tabela de sapatos como uma chave externa.

Quando estiver acabado, desenhe os vínculos.

id_sapato	nome_sapato	id_mulher
1	Sandália de Laço Manolo	3
2	Tamancos de Crocodilo	2
3	Chinelos Old Navy	1
4	Botas Prada	1
5	Tamancos de Crocodilo	3
6	Chinelos Old Navy	3
7	Botas Prada	3
8	Sandália de Laço Manolo	4
9	Chinelos Old Navy	4
10	Botas Prada	4

id_mulher	mulher
1	Carrie
2	Samantha
3	Charlotte
4	Miranda



## Relações de dados: nós precisamos de uma junção de tabelas.

Se você mesmo pode descobrir, ao adicionar as chaves primárias de duas tabelas como chaves estrangeiras na outra, são gerados dados duplicados em nossa tabela. Perceba quantas vezes os nomes das mulheres reaparecem. Devíamos vê-las apenas uma vez.

Nós precisamos de uma tabela para dar um passo entre estas duas tabelas com relacionamento muitos-para-muitos e muitas para um-para-muitos. Esta tabela armazenará todos os valores id\_mulher bem como os valores id\_sapato. Nós precisamos o que é chamado **junção de tabelas (tabela de conexão)**, que conterá a chave primária das duas tabelas que queremos relacionar.

Vincular estas duas tabelas diretamente uma com a outra não vai dar certo porque acabamos com dados duplicados graças aos relacionamentos muitos-para-muitos.

<b>id_mulher</b>	<b>mulher</b>
1	Carrie
2	Samantha
3	Charlotte
4	Miranda



<b>id_sapato</b>	<b>nome_sapato</b>
1	Sandália de Laço Manô
2	Tamancos de Crocodilo
3	Chinelos Old Navy
4	Botas Prada

um-para-muitos

Pegue a chave primária daqui...

...e a chave primária daqui...

...e coloque ambas na junction table.

um-para-muitos

A tabela de conexão contém as chaves primárias das duas tabelas que você quer relacionar.

Dai você vai precisar vincular as colunas das chaves primárias de cada uma das tabelas originais com as colunas correspondentes na tabela de conexão.

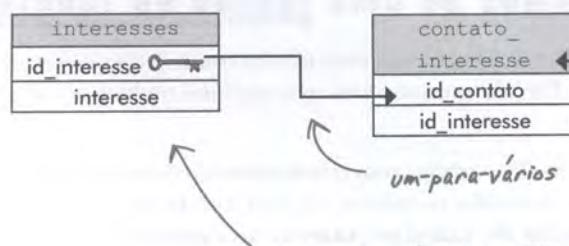
<b>id_mulher</b>	<b>id_sapato</b>
1	3
1	4
2	1
3	1
3	2
3	3
3	4
4	1
4	3
4	4

Muitos-para-muitos: uma tabela de conexão armazena a chave de cada tabela.

## Padrões de dados: muitos-para-muitos

Agora você sabe o segredo dos relacionamentos muitos-para-muitos - é geralmente feito de dois relacionamentos um-para-muitos, com uma **tabela de conexão no meio**. Precisamos associar **UMA** pessoa na tabela meus\_contatos com **VÁRIOS** interesses na nossa nova tabela de interesses. Todavia cada valor da coluna interesses pode apontar para mais de uma pessoa, então este relacionamento se encaixa no padrão **muitos-para-muitos**.

A coluna interesses pode ser convertida em vários relacionamentos do tipo muitos-para-muitos utilizando este esquema. Cada pessoa pode ter mais que um interesse, e para cada interesse pode haver mais de uma pessoa que o tem em comum:



<b>meus_contatos</b>
<b>id_contato</b>
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
profissao
cidade
estado
estado_civil
procura

Estas duas tabelas, meus\_contatos e interesses, agora possuem um relacionamento recíproco do tipo muitos-para-muitos.

## não existem Perguntas Idiotas

**P:** Eu tenho que criar a tabela do meio quando tenho relacionamentos muitos-para-muitos?

**R:** Sim, deveria. Se você tem tabelas com relacionamento muitos-para-muitos entre duas tabelas, vai acabar tendo grupos repetidos, violando a primeira forma normal. (Um lembrete sobre normalização está por vir nas próximas páginas.)

Há uma boa razão para violar a primeira forma normal, e muitas boas razões para não o fazer. O grande problema é que você terá dificuldade ao fazer consultas em sua tabela com vários dados repetidos.

**P:** Qual a vantagem de alterar minha tabela desta forma? Eu poderia apenas colocar todos os interesses em uma tabela `meus_contatos` e `nome_interesse`. Eu teria dados repetidos, mas, além disso, por que não?

**R:** Você definitivamente verá uma vantagem quando começar a fazer consultas nestas tabelas múltiplas com conexões no próximo capítulo. Ela vai poder te ajudar também, dependendo de como utilizará seus dados. Você poderá ter uma tabela onde seu maior interesse será nas conexões muitos-para-muitos do que nos dados contidos em qualquer uma das tabelas.

**P:** E se ainda assim eu não me importar com dados repetidos?

**R:** Combinar tabelas ajuda a preservar a integridade de seus dados. Se você tiver que apagar alguém de `meus_contatos`, nunca vai tocar na tabela `interesses`, apenas na tabela `contato_interesse`. Sem a tabela separada, você poderia apagar registros errados. É mais seguro assim.

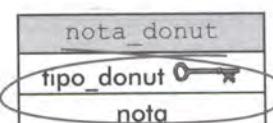
E quando o assunto é atualizar informações, também é ótimo. Suponha que tenha digitado errado o nome de um estranho hobby, como "espeluncologia". Quando consertá-lo, você fará isso em apenas uma linha na tabela `interesses`, e nunca terá que tocar nas tabelas `contato_interesse` ou `meus_contatos` para arrumar isso.

## — NOMEIE AQUELE RELACIONAMENTO —

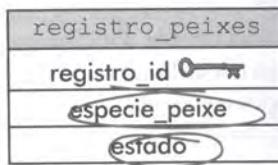
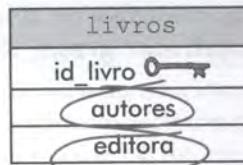
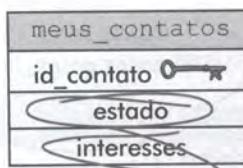
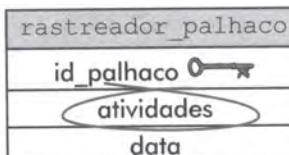
Em cada um dos pedaços de tabelas abaixo, decida se cada uma das colunas circuladas é melhor representada por relacionamentos um-para-muitos ou muitos-para-muitos.

(Lembre-se que se for um-para-muitos ou muitos-para-muitos, a coluna seria puxada da tabela e vinculada a um campo ID).

### COLUNA



### RELACIONAMENTO

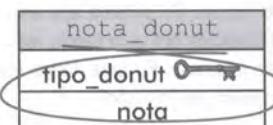


# NOMEIE AQUELE RELACIONAMENTO

Em cada um dos pedaços de tabelas abaixo, decida se cada uma das colunas circuladas é melhor representada por relacionamentos um-para-muitos ou muitos-para-muitos.

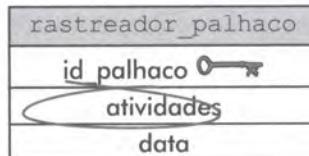
(Lembre-se que se for um-para-muitos ou muitos-para-muitos, a coluna seria puxada da tabela e vinculada a um campo ID).

## COLUNA

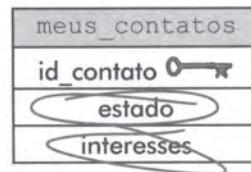


## RELACIONAMENTO

um-para-muitos

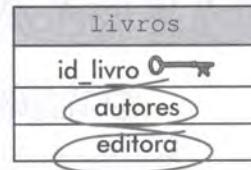


muitos-para-muitos



um-para-muitos

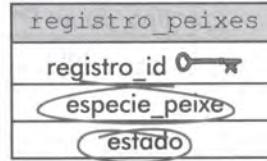
muitos-para-muitos



Este aqui é complicado, mas já que um livro pode ter mais que um autor, então é do tipo muitos-para-muitos.

muitos-para-muitos

um-para-muitos



um-para-muitos

um-para-muitos

## Padrões de dados: consertar

Eu sei onde você está indo  
depois daqui. Nós vamos alterar o banco  
de dados gregs\_list e meus\_contatos para  
um formato multi-tabelas, certo?

**Quase. Agora que já sabe sobre os padrões de dados, estamos quase prontos para rearquitetar gregs\_list.**

Sabemos que a coluna interesses pode ser alterada para um relacionamento um-para-muitos com outra tabela. E também precisamos consertar a coluna procura da mesma forma. Estas alterações nos deixarão aptos na **primeira forma normal\***.

Entretanto, não podemos parar na primeira forma normal. Precisamos ir mais longe com a normalização. Quanto mais normalizarmos a tabela agora, mas fácil será para você acessar seus dados com consultas e, no próximo capítulo, com conexões. Antes de criarmos um novo esquema para gregs\_list, vamos pegar um desvio para aprender mais sobre níveis de normalização.



\*Você pode se sentir um pouco coagido a revirar as páginas para alguns capítulos anteriores para refrescar sua memória sobre a primeira forma normal. Não será necessário, falaremos sobre ela na próxima página.

## Sem estar na primeira forma normal

Já falamos sobre a primeira forma normal. Vamos dar uma olhada nisso novamente, e então levar nossa normalização para mais longe, para a Segunda e até a Terceira Forma Normal.

Antes de irmos, vamos recapitular apenas o que é que deixa a tabela em 1NF.

meus_contatos
id_contato
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
profissao
cidade
estado
estado_civil
interesses
procura

## Primeira Forma Normal, ou 1NF:

Regra 1: Colunas contêm apenas dados atômicos

Regra 2: Não há grupos de dados repetidos

Tabelas abaixo não estão em Primeira Forma Normal. Note que a segunda tabela teve colunas extras de cor adicionadas, mas cores em si repetem-se uma vez por linha na nova tabela:

### Não está em 1NF

Id_brinquedo	brinquedo	cor
5	wiffleball	branco, amarelo, azul
6	frisbee	verde, amarelo
9	pipa	vermelha, azul, verde
12	ioiô	branco, amarelo

Para serem atômicas, as cores das colunas deveria conter apenas uma das cores e não duas ou três cores na mesma coluna.

### Ainda não está em 1NF

Id_brinquedo	brinquedo	cor1	cor2	cor3
5	wiffleball	branco	amarelo	azul
6	frisbee	verde	amarelo	
9	pipa	vermelha	azul	verde
12	ioiô	branco	amarelo	

Esta tabela ainda não está na 1NF porque as próprias colunas estão armazenando a mesma categoria de dados, todos VARCHAR com uma cor do brinquedo.

# Finalmente 1NF

Dê uma olhada no que a gente fez aqui:

## In 1NF

Chave Primária.

Id_brinquedo	brinquedo
5	wiffleball
6	frisbee
9	pipa
12	ioiô

Agora que sabemos que estas duas tabelas estão combinadas pelas chaves primárias, não precisamos ficar desenhando linhas conectadoras para todo lado.

Chave Externa.

Id_brinquedo	cor
5	branco
5	amarelo
5	azul
6	verde
6	amarelo
9	vermelha
9	azul
9	verde
12	branco
12	amarelo

Nenhuma das linhas está repetida dentro da coluna. Há uma cor por linha e cada linha é única.

Nós podemos utilizar tanto o id\_brinquedo e valor para conjuntos para formar uma chave primária única.

Se adicionarmos a id\_brinquedo em uma tabela separada como uma chave externa estará ótimo porque os valores que ela armazena não precisam ser únicos. E se também adicionássemos valores de cor para aquela tabela, **todas as linhas são únicas** porque cada cor PLUS (MAIS) cada id\_brinquedo juntos fazem uma **combinação única**.



Uma chave primária multi-coluna? Mas a chave primária não precisa estar em apenas uma tabela?

**Não. Uma chave feita de duas ou mais colunas é conhecida como uma chave composta.**

Vamos dar uma olhada em como isto funciona em mais algumas tabelas.

## Chaves compostas utilizam múltiplas colunas

Até agora falamos sobre como os dados em uma tabela se relacionam com outras tabelas (um-a-um, um-para-muitos). O que não levamos em conta é como as **colunas em uma tabela se relacionam umas com as outras**. Entendimento é a chave para entender a segunda e a terceira forma normal.

E depois de entendido isto, podemos criar esquemas de bancos de dados que farão consultas a tabelas múltiplas muito mais fáceis.

Então o que é exatamente uma chave composta?

Você vai querer tabelas bem projetadas quando estivermos falando sobre conexões no próximo capítulo!

**UMA CHAVE COMPOSTA** é uma chave primária composta de múltiplas colunas criando uma chave única.



Suponha esta tabela de super-heróis. Ela não tem uma chave primária, mas nós podemos criar uma chave composta a partir das colunas nome e poder. Há alguns nomes duplicados entre nome e poder, coloque-os juntos, e o conjunto das duas colunas terá um valor único.

Nós poderíamos criar esta tabela e designar estes dois campos para serem a chave primária composta. Estamos presumindo que nunca teremos exatamente os mesmos nomes e poderes, sendo então únicos.

super\_herois

nome	poder	fraqueza
Super Lixeiro	Limpa rapidamente	alvejante
O Corretor	Faz dinheiro do nada	NULL
Super Cara	Voa	pássaros
Garçom Maravilha	Nunca esquece um pedido	insetos
Homem Poeira	Cria tempestades de areia	alvejante
Super Cara	Super força	o outro Super Cara
Mulher Fúriosa	Fica muito, muito, muito nervosa	NULL
O Sapo	Língua da Justiça	insetos
Garota Bibliotecária	Pode achar qualquer coisa	NULL
Mulher Ganso	Voa	NULL
Super Palito	Substitui humanos	Jogo da Força



Super Palito, Super Palito, faz tudo que um homem não pode. Tudo que precisa é de um n.º 2 para dizer ao Super Palito o que fazer. Solte sua imaginação. Vá desenhar seu próprio Super Palito!

## Por que os super-heróis podem ser dependentes

Os super-heróis têm estado ocupados! Aqui está a tabela super\_herois. Nós estamos na 1NF, mas há outro problema.

Por exemplo, a coluna iniciais contém as letras iniciais do valor na coluna nome. O que aconteceria se um Super-herói mudasse seu nome? Instantaneamente, a coluna iniciais também seria alterada. A coluna iniciais é conhecida como **funcionalmente dependente** da coluna nome.

Por que esses dois nomes idênticos, com a coluna poder igual, não conseguem para criar uma chave primária verdadeiramente única?

Quando os dados de uma coluna devem mudar, quando os dados de outra coluna são modificados, a primeira coluna é funcionalmente dependente da segunda.

super\_herois

nome	poder	fraqueza	cidade	país	arquiinimigo	iniciais
Super Lixeiro	Limpa rapidamente	alvejante	Gotham	US	Verminador	SL
O Corretor	Faz dinheiro do nada	NULL	Nova York	US	Senhor Impostos	OC
Super Cara	Voa	pássaros	Metrópolis	US	Super Colega	SC
Garçom Maravilha	Nunca esquece um pedido	insetos	Paris	França	Garota Coma Tudo O Que Puder	GM
Homem Poeira	Cria tempestades de areia	alvejante	Tulsa	US	Hoover	HP
Super Cara	Super força	alumínio	Metrópolis	US	Badman	SC
Mulher Fúriosa	Fica muito, muito, muito nervosa	NULL	Roma	Itália	O Terapeuta	MF
O Sapo	Língua da Justiça	insetos	Londres	Inglaterra	Heron	OS
Garota Bibliotecária	Pode achar qualquer coisa	crianças	Springfield	US	Horror do Caos	GB
Mulher Ganso	Voa	NULL	Mineápolis	US	O Desistente	MG
Super Palito	Substitui humanos	Jogo da Força	Londres	Inglaterra	Homem Borracha	SP



## Aponte seu lápis

Agora que já sabe que a coluna iniciais é dependente da coluna nome na tabela super\_herois. Você consegue ver qualquer outra dependência. Escreva-as aqui.

.....  
.....  
.....



## Aponte seu lápis Solução

Agora que já sabe que a coluna iniciais é dependente da coluna nome na tabela super\_herois. Você consegue ver qualquer outra dependência. Escreva-as aqui.

A Iniciais é dependente da coluna nome ←  
Fraguezas é dependente da coluna nome ←  
Arquiinimigo é dependente da coluna nome ←  
Cidade é dependente da coluna país ←

Estas não mencionam de quais tabelas as colunas são. O que vai importar quando você adicionar mais tabelas. Há uma forma de atalho para indicar as dependências e as tabelas de onde elas pertencem.



Volta pelos b

## Notas de atalho

Uma forma rápida para descrever uma dependência funcional é escrevendo desta forma:

**T . x →; T . y ←** O termo técnico para isso é nota

Que pode ser lida assim: "na tabela relacional chamada T, a coluna y é dependente funcional da coluna x".

Basicamente, você as lê da direita para esquerda para ver o que é dependente funcionalmente do que.

Vamos ver isto aplicado a nossa tabela super\_herois:

**super\_herois.nome →; super\_herois.iniciais**

"Na tabela relacional super\_herois, a coluna iniciais é dependente funcional da coluna nome".

**super\_herois.nome →; super\_herois.fraqueza**

"Na tabela relacional super\_herois, a coluna fraqueza é dependente funcional da coluna nome".

`super_herois.nome ->; super_herois.arquiinimigo`

"na tabela relacional `super_herois`, a coluna `arquiinimigo` é dependente funcional da coluna `nome`".

`super_herois.pais ->; super_herois.cidade`

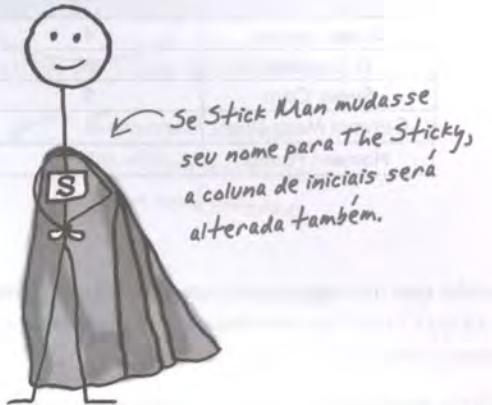
"na tabela relacional `super_herois`, a coluna `cidade` é dependente funcional da coluna `país`".

## Dependências de super-heróis

se nosso super-herói fosse mudar seu nome, a coluna `iniciais` também será alterada também, fazendo-a **dependente** da coluna `nome`.

se nosso arquiinimigo decidir mudar seu lar para uma nova cidade, o local será alterado, mas nada mais será. Isto faz da coluna `arquiinimigo_cidade` na tabela abaixo completamente **independente**.

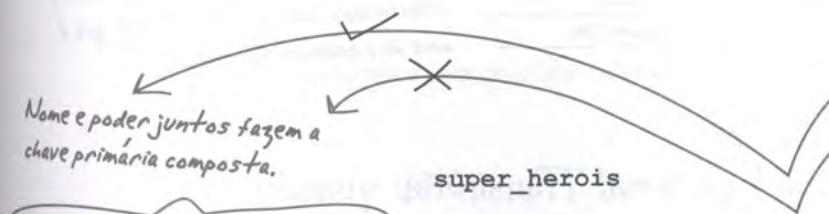
uma coluna **dependente** é qualquer uma que **contenha dados que poderiam ser alterados se outra coluna fosse alterada**. Colunas não-dependentes ficam sós.



## Dependência Funcional Parcial

uma dependência funcional parcial quer dizer uma coluna que não é chave primária em todas as colunas, mas depende de algumas, mas não todas as colunas em uma chave primária composta.

Na nossa tabela `super_herois`, a coluna `iniciais` é **parcialmente dependente** no nome, porque se o nome na tabela `super_herois` muda, os valores da coluna `iniciais` também mudarão, mas se a coluna `poder` é alterada, e só o nome, as iniciais de nosso super-herói serão as mesmas.



A `iniciais` depende da coluna `nome`, mas não da coluna `poder`, então esta tabela contém uma dependência funcional parcial.

0+*	0+*	fraqueza	cidade	iniciais	país	arquiinimigo
Super Lixeiro	Limpa rapidamente	alvejante	Gotham	SL	US	Verminador
O Corretor	Faz dinheiro do nada	NULL	Nova York	OC	US	Senhor Impostos
Super Cara	Voa	pássaros	Metrópolis	SC	US	Super Colega
Garota Maravilha	Nunca esquece um pedido	Insetos	Paris	GM	França	Garota Coma Tudo O Que Puder
Homem Poeira	Cria tempestades de areia	alvejante	Tulsa	HP	US	Hoover
Super Cara	Super força	alumínio	Metrópolis	SC	US	Badman
Mulher Furiosa	Fica muito, muito, muito nervosa	NULL	Roma	MF	Itália	O Terapeuta
O Sapo	Língua da Justiça	Insetos	Londres	OS	Inglaterra	Heron
Garota Bibliotecária	Pode achar qualquer coisa	crianças	Springfield	GB	US	Horror do Caos
Mulher Ganso	Voa	NULL	Mineápolis	MG	US	O Desistente
Super Palito	Substitui humanos	Jogo da Força	Londres	SP	Inglaterra	Homem Borracha

## Dependência Funcional Transitória

Você precisa levar em consideração como cada coluna não-chave se relaciona com as outras. Se um arquiinimigo se muda para uma outra cidade, isto não altera sua `id_arquiinimigo`.

nome	id_arquiinimigo	arquiinimigo_cidade
Super Lixeiro	4	Kansas City
O Corretor	8	Newark
Super Cara	5	Metrópolis
Garçom Maravilha	1	Paris
Homem Poeira	2	Kansas City

Suponha que um super-herói muda seu arquiinimigo. O `id_arquiinimigo` mudaria, e isto **poderia** alterar a coluna `arquiinimigo_cidade`.

Se alterar qualquer das colunas não-chaves pode levar a qualquer uma das outras colunas a serem alteradas. Você tem uma **dependência transitória**.

Se alterar qualquer das colunas não-chaves pode levar a qualquer uma das outras colunas a serem alteradas.  
Você tem uma dependência transitória.

nome	id_arquiinimigo	arquiinimigo_cidade
Super Lixeiro	2	Kansas City
O Corretor	8	Newark
Super Cara	5	Metrópolis
Garçom Maravilha	1	Paris
Homem Poeira	2	Kansas City

Isto é chamado de dependência funcional transitória porque a coluna não-chave `arquiinimigo_cidade` é relacionada com `id_arquiinimigo`, que é uma das colunas não-chave.

Dependência Funcional Transitória: quando qualquer coluna não-chave é relacionada a qualquer outra das colunas não-chave.



Dê uma olhada nesta tabela listando títulos de livros. A coluna id\_edit identifica a editora. Já a coluna cidade\_pub é a cidade onde o livro foi publicado.

### Exercícios

autor	título	copyright	id_edit	cidade_pub
John Deere	Easy Being Green	1930	2	Nova York
Fred Mertz	I Hate Lucy	1968	5	Boston
Lassie	Help Timmy!	1950	3	São Francisco
Timmy	Lassie, Calm Down	1951	1	Nova York

Escreva o que acontecerá ao valor na coluna copyright se o título do livro na terceira linha for alterado para: 'Help Timmy! I'm Stuck Down a Well'

Se o título é alterado, o valor copyright também mudará.

Copyright depende  
do título, então seu  
valor será alterado.

O que acontecerá com o valor na coluna copyright se o autor do livro na terceira linha mudar para 'Rin Tin Tin', mas o título permanecer o mesmo?

O que aconteceria com 'Easy Being Green' se alterássemos seu id\_edit para 1?

O que aconteceria ao valor de id\_edit de 'I Hate Lucy' se a editora se mudasse para Sebastopol?

O que aconteceria com o valor de cidade\_pub de 'I Hate Lucy' se alterássemos o valor de id\_edit para 1.



## Solução dos Exercícios

Dê uma olhada nesta tabela listando títulos de livros. A coluna `id_edit` identifica a editora. Já a coluna `cidade_pub` é a cidade onde o livro foi publicado.

Escreva o que acontecerá ao valor na coluna `copyright` se o título do livro na terceira linha for alterado para: 'Help Timmy! I'm Stuck Down a Well'

*Se o título é alterado, o valor copyright também mudará.*

*Copyright depende do título, então seu valor será alterado.*

O que acontecerá com o valor na coluna `copyright` se o autor do livro na terceira linha mudar para 'Rin Tin Tin', mas o título permanecer o mesmo?

*Autor e título juntos fazem a chave primária composta.*

*Se o autor mudar, e não o título, o copyright também muda.*

*Copyright depende do título. Também depende do autor.*

autor	título	copyright	id_edit	cidade_pub
John Deere	Easy Being Green	1930	2	Nova York
Fred Mertz	I Hate Lucy	1968	5	Boston
Lassie	Help Timmy!	1950	3	São Francisco
Timmy	Lassie, Calm Down	1951	1	Nova York

O que aconteceria com 'Easy Being Green' se alterássemos seu `id_edit` para 1?

*id\_edit é independente de cidade\_pub, então id\_edit continuaria a mesma.*

*A cidade\_pub não mudará*

*A cidade\_pub para id\_edit 1 e id\_edit 2 é Nova York, então a cidade não mudará (ainda que cidade\_pub é dependente transitoriamente de id\_edit).*

O que aconteceria ao valor de `id_edit` de 'I Hate Lucy' se a editora se mudasse para Sebastopol?

*O id\_edit continuaria o mesmo*

*cidade\_pub é dependente transitório de id\_edit então o valor da coluna cidade é alterado.*

O que aconteceria com o valor de `cidade_pub` de 'I Hate Lucy' se alterássemos o valor de `id_edit` para 1.

*O cidade\_pub mudaria para Nova York.*

*Cidade\_pub é dependente do valor contido na coluna id\_edit. Tampouco a coluna é coluna-chave, então esta é uma dependência funcional transitória.*

autor	título	copyright	id_edit	cidade_pub
John Deere	Easy Being Green	1930	2	Nova York
Fred Mertz	I Hate Lucy	1968	5	Boston
Lassie	Help Timmy!	1950	3	São Francisco
Timmy	Lassie, Calm Down	1951	1	Nova York

## não existem Perguntas Idiotas

**Q:** Há algum jeito fácil de se evitar uma dependência funcional parcial?

**A:** Utilizando um campo id em meus\_contatos permite que você evite este problema por completo. Já que é uma nova chave que existe apenas na indexar aquela tabela, nada é dependente dela.

**Q:** Então, a não ser que quando eu criar conexões para tabelas, por que eu iria querer criar uma chave composta para colunas na minha tabela? Por que não sempre criar um campo id?

**A:** É de fato uma maneira de agir, mas você vai encontrar argumentos convincentes para ambos os lados se pesquisar na web por "chave sintética ou natural". Você também vai encontrar debates acalorados. Deixaremos você se decidir sobre o assunto. Neste livro, vamos nos ater primariamente a campo com chave primária única e sintética para manter a sintaxe mais simples para que você aprenda os conceitos e não fique atolado com as implementações.



Olhe, estas dependências são legais e tal, mas o que elas têm a ver com mover de primeira forma normal para segunda forma normal?

### **Adicionar colunas de chave primária nas nossas tabelas está nos ajudando a alcançar a 2NF.**

Com o objetivo de facilitar, e garantir unicidade, temos geralmente adicionado colunas para todas as nossas tabelas para agirem como chaves primárias. Isto de fato nos ajuda a alcançar a 2NF, porque **a segunda forma normal foca em como a chave primária em uma tabela se relaciona com os dados inseridos na mesma tabela.**

## Segunda Forma Normal

Levemos em conta duas tabelas existentes para acompanhar um inventário de brinquedos para nos ajudar a melhor compreender como a segunda forma normal se foca no relacionamento entre chaves primárias de tabelas e os dados inseridos nela.

id_bringuedo	brinquedo
5	wiffleball
6	frisbee
9	pipa
12	ioiô

<i>Chave composta.</i>				
id_bringuedo <i>0 + R</i>	id_loja <i>0 + R</i>	cor	inventario	endereco_loja
5	1	branco	34	23 Maple
5	3	amarelo	12	100 E. North St.
5	1	azul	5	23 Maple
6	2	verde	10	1902 Amber Ln.
6	4	amarelo	24	17 Engleside
9	1	vermelho	50	23 Maple
9	2	azul	2	1902 Amber Ln.
9	2	verde	18	1902 Amber Ln.
12	4	branco	28	17 Engleside
12	4	amarelo	11	17 Engleside

Há muitas repetições nessa coluna. Elas realmente não têm nada a ver com o inventário; elas estão relacionadas com a loja.

Podemos ter que repensar nesta coluna. Ela pertence a uma tabela de brinquedos do que a uma tabela de inventário. Nossa coluna id\_bringuedo deve ser o suficiente para identificar cor do bringuedo e tipo de bringuedo.

Inventário é dependente de ambas as colunas que fazem a chave composta, então não tem uma dependência funcional parcial.

Note como o `endereco_loja` é repetido quando um brinquedo é associado com a `id_loja`. Se precisarmos do `endereco_loja`, teremos que fazer em cada linha onde há referência a ela na tabela. Quanto mais linhas são atualizadas ao longo do tempo, maior a possibilidade para que erros atormentem seus dados.

Se puxarmos a coluna `endereco_loja` para outra tabela, teríamos que fazer apenas uma alteração.

## Nós já devemos ser 2NF agora...

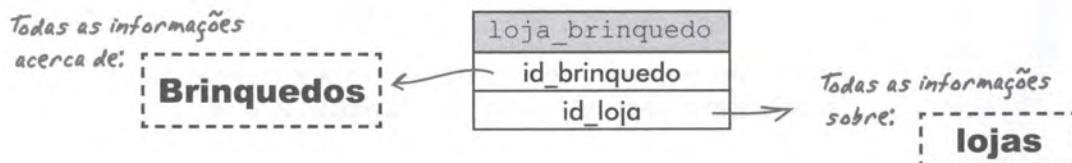
**Uma tabela em 1NF é também 2NF se todas as colunas na tabela são partes da chave primária.**

Nós poderíamos criar uma nova tabela com uma chave primária composta com `id_brinquedo` e `id_loja`. Então teríamos uma tabela com todas as informações dos brinquedos e uma tabela com todas as informações sobre as lojas, com nossa nova tabela conectando-as.

**Sua tabela 1NF é também 2NF se todas as colunas na tabela são partes da chave primária.**

**OU**

**Tem uma só coluna como chave primária.**



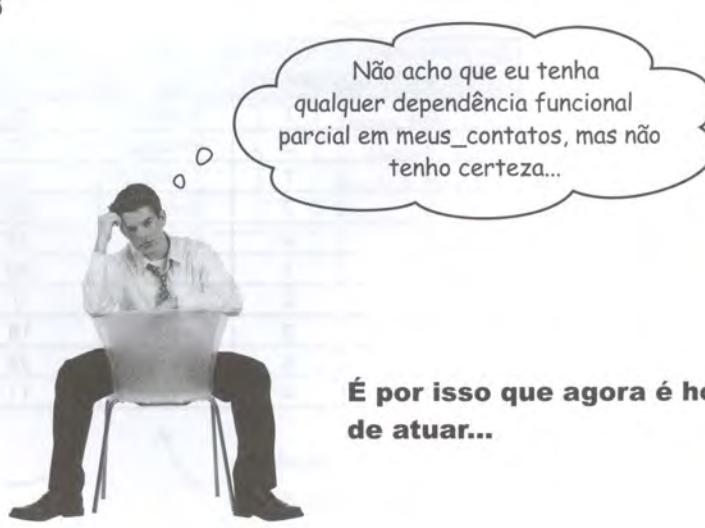
**Sua tabela 1NF é também 2NF se ela tem uma só coluna como chave primária.**

Esta é uma ótima razão para atribuir uma coluna `id` com `AUTO_INCREMENT`.

**Segunda Forma Norma ou 2NF:**

**Regra 1: Ser 1NF**

**Regra 2: Não ter dependências funcionais parciais.**



## Seja uma tabela 2NF sem dependências funcionais parciais



Seu trabalho agora é atuar como se fosse uma tabela, e remover todas as dependências funcionais parciais de você. Olhe para cada tabela no diagrama abaixo e desenhe linhas através das colunas que seriam melhores se movidas para uma outra tabela.

Estas duas fazem uma chave primária composta única.

inventario_brinquedo
id_brinquedo
id_loja

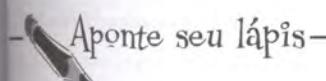
venda_biscoito
quantidade
id_garota
data
nome_garota
lider_tropa
total_venda

cantores
id_cantor
sobrenome
primeiro_nome
agencia
agencia_estado

filmes
id_filme
titulo
genero
alugado_por
data_devolucao
censura

salario
id_empregado
sobrenome
primeiro_nome
salario
gerente
empregado_email
data_contrato

raca_caes
raca
descricao
peso_medio
altura_media
id_clube
estado_clube



Aponte seu lápis ————— Reestruture estas tabelas em três tabelas que sejam 2NF.

Uma conterá informação sobre brinquedo, uma conterá informação sobre loja, e a terceira conterá o inventário e se conecta com as outras duas. Dê nomes significativos para as três.

Finalmente, adicione estas colunas adicionais nas tabelas apropriadas:

id_brinquedo	brinquedo
5	whiffleball
6	frisbee
9	pipa
12	ioiô

id_brinquedo	id_loja	cor	inventario	endereco_loja
5	1	branco	34	23 Maple
5	3	amarelo	12	100 E. North St.
5	1	azul	5	23 Maple
6	2	verde	10	1902 Amber Ln.
6	4	amarelo	24	17 Engleside
9	1	vermelho	50	23 Maple
9	2	azul	2	1902 Amber Ln
9	2	verde	18	1902 Amber Ln
12	4	branco	28	17 Engleside
12	4	amarelo	11	17 Engleside

# Seja uma tabela 2NF sem dependências funcionais parciais



Seu trabalho agora é atuar como se fosse uma tabela, e remover todas as dependências funcionais parciais de você. Olhe para cada tabela no diagrama abaixo e desenhe linhas através das colunas que seriam melhores se movidas para uma outra tabela.

*Chave primária*

cantores
id_cantor
sobrenome
primeiro_nome
agencia
agencia_estado

*Enquanto estas duas devem ser puxadas de uma tabela agência (porque duas agências podem ter o mesmo nome), ela não tem dependência funcional parcial.*

*Chave primária*

*Enquanto estas devem ser retiradas desta tabela, elas não são dependência funcional parcial.*

salario
id_empregado
sobrenome
primeiro_nome
salario
gerente
empregado_email
data_contrato

*Estas duas fazem uma chave primária composta única.*

inventario_bringuedo
id_bringuedo
id_loja

venda_biscoito
quantidade
id_garota
data
nome_garota
lider_tropa
total_venda

*Depois de removermos estas colunas para formar as restantes podem formar uma chave primária composta.*

filmes
id_filme
titulo
genero
alugado_por
data_devolucao
censura

*Chave primária*

*Estas colunas possuem somente dependência funcional transitória.*

raca_caes
raca
descricao
peso_medio
altura_media
id_clube
estado_clube

*Chave primária composta*

*id\_clube pode pertencer a esta tabela (se este for um relacionamento um-a-um), mas estado\_clube não faz parte da. Mesmo assim, nenhuma das colunas é dependente funcional parcial.*

Aluno 12	12	Matheus
Aluno 13	13	Juliana
Aluno 14	14	Paulo
Aluno 15	15	Carolina
Aluno 16	16	Thiago
Aluno 17	17	Adriana
Aluno 18	18	Renata
Aluno 19	19	Isabella
Aluno 20	20	Diego


**Aponte seu lápis**  
**Solução**

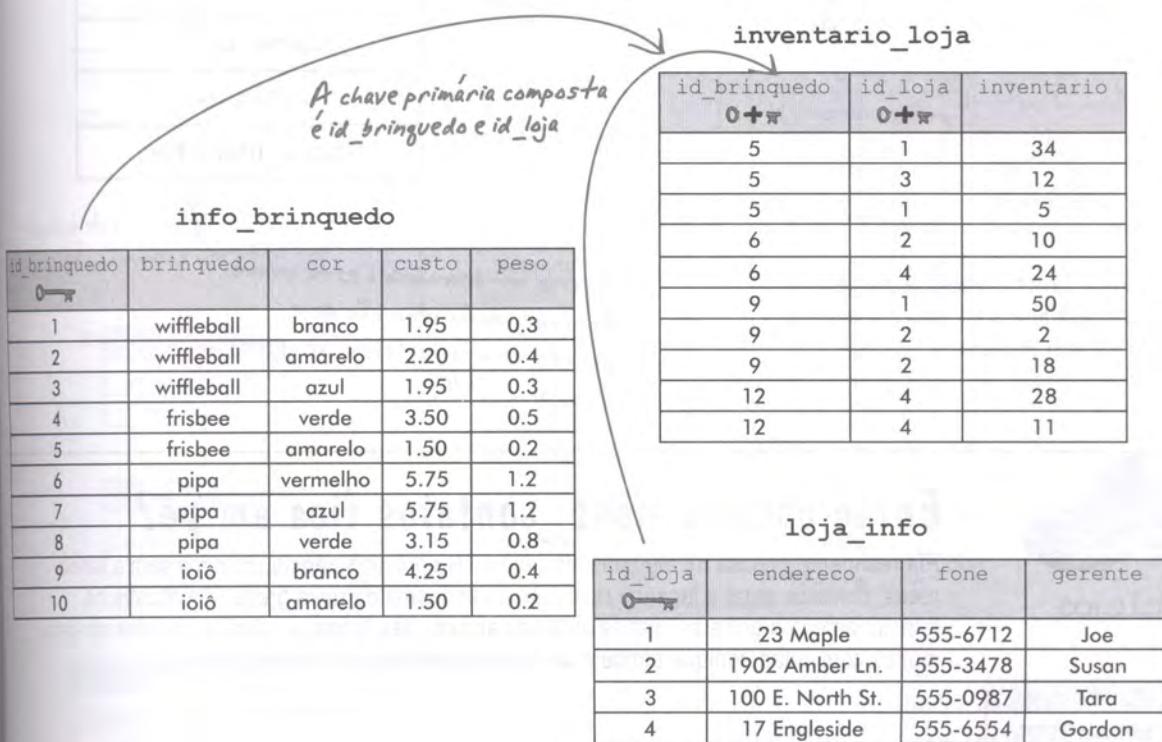
Reestruture estas tabelas em três tabelas que sejam 2NF.

Uma conterá informação sobre brinquedo, uma conterá informação sobre loja, e a terceira conterá o inventário e se conecta com as outras duas. Dê nomes significativos para as três.

Finalmente, adicione estas colunas adicionais nas tabelas apropriadas:

id_brinquedo	brinquedo
5	wiffleball
6	frisbee
9	pipa
12	ioiô

id_brinquedo	id_loja	cor	inventario	endereco_loja
5	1	branco	34	23 Maple
5	3	amarelo	12	100 E. North St.
5	1	azul	5	23 Maple
6	2	verde	10	1902 Amber Ln.
6	4	amarelo	24	17 Engleside
9	1	vermelho	50	23 Maple
9	2	azul	2	1902 Amber Ln.
9	2	verde	18	1902 Amber Ln.
12	4	branco	28	17 Engleside
12	4	amarelo	11	17 Engleside



## Terceira Forma Normal (finalmente)

Devido ao fato de, geralmente, adicionarmos chaves primárias artificiais, conseguir com que nossas tabelas alcancem a segunda forma normal não é um problema para nós. Qualquer tabela com uma **chave primária artificial** e sem chaves primárias compostas é sempre uma 2NF.

Devemos, no entanto, nos certificar de que estamos na 3NF.

### Terceira Forma Normal ou 3NF:

**Regra 1: Seja uma 2NF**

**Regra 2: Não tem dependências transitórias.**

Considere o que aconteceria se mudássemos um valor em qualquer destas três colunas: nome\_curso, instrutor e fone\_instrutor.

- ⇒ Se alterarmos nome\_curso, a coluna instrutor e instrutor\_fone não serão alteradas.
- ⇒ Se alterarmos a coluna fone\_instrutor, a instrutor e a nome\_curso não precisam ser alteradas.
- ⇒ Se alterarmos o instrutor, a fone\_instrutor será alterada. Nós encontramos nossa dependência transitória.

Nós podemos ignorar a chave-primária ao considerar a 3NF.

cursos
id_curso
nome_curso
instrutor
fone_instrutor

Deveria ser bem óbvio a esta altura que fone\_instrutor não deve pertencer a esta tabela, se queremos alcançar a 3NF.



Exercícios

### Então como a meus\_contatos fica em pé?

Ela realmente precisa de algumas alterações. Na próxima página, comece com a tabela meus\_contatos atual e faça um rascunho do esquema da nova gregs\_list. Mostre os relacionamentos entre as chaves externas através das linhas, e relacionamentos um-para-muitos com setas. Indique também as chaves primárias ou chaves compostas.

meus_contatos
id_contato
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
profissao
cidade
estado
estado_civil
interesses
procura

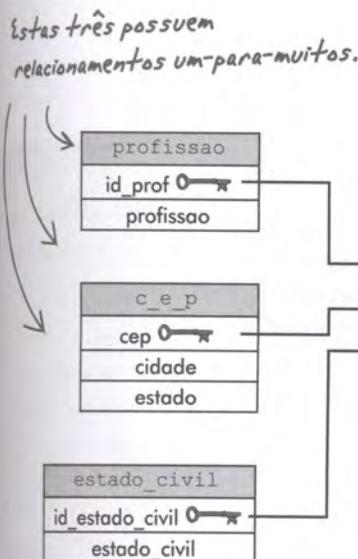
Dica: na nossa versão que se encontra na próxima página, temos 8 tabelas (Nós adicionamos uma coluna para CEP. Antes disso, eram 7 colunas)



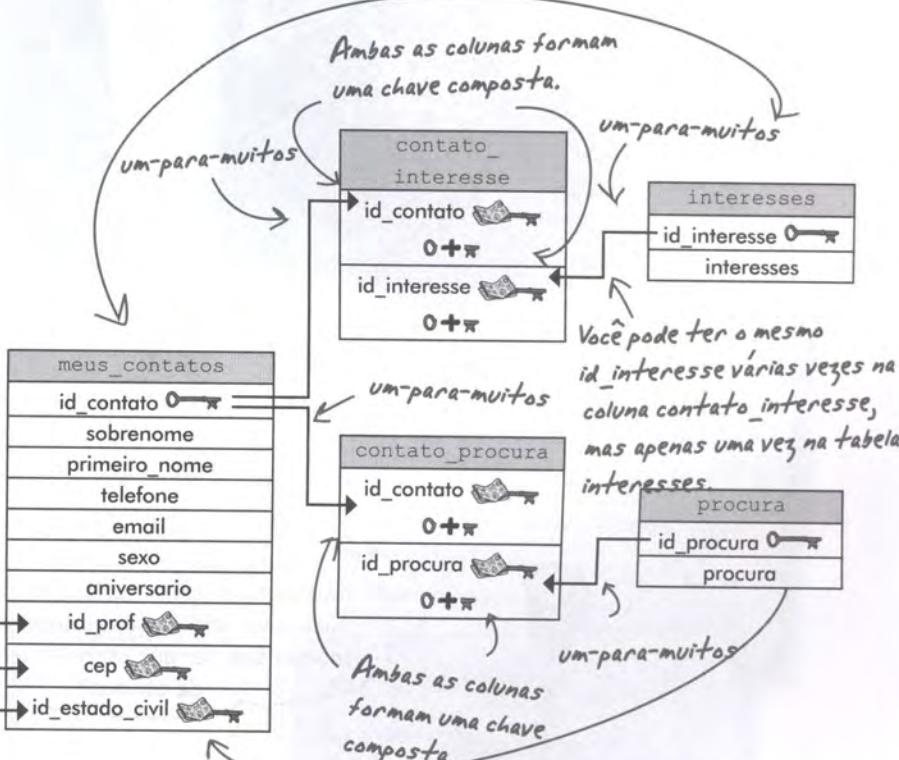
## Então como a meus\_contatos fica em pé?

Ela realmente precisa de algumas alterações. Na próxima página, comece com a tabela meus\_contatos atual e faça um rascunho do esquema da nova gregs\_list. Mostre os relacionamentos entre as chaves externas através das linhas, e relacionamentos um-para-muitos com setas. Indique também as chaves primárias ou chaves compostas.

meus_contatos
id_contato 0→*
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
profissao
cidade
estado
estado_civil
interesses
procura



Este é um relacionamento muitos-para-muitos, que é feito de dois relacionamentos um-para-muitos e uma tabela de conexão.



Este é um relacionamento muitos-para-muitos, que é feito de dois relacionamentos um-para-muitos e uma tabela de conexão.

## E, então, Regis (e gregs\_list) viveram felizes para sempre

Greg é capaz de encontrar o par perfeito para Regis utilizando seu novo e normalizado banco de dados. Melhor ainda, ele também é capaz de encontrar facilmente pares para mais amigos, mantendo o sonho da Lista de Greg vivo.



Fim



Não tão rápido! Agora eu tenho que consultar todas estas novas tabelas e uni-las manualmente! Como chego aos meus dados agora como todas estas tabelas sem ter que escrever uma centena de colunas?

**É aqui que as conexões entram em ação.**

Te vejo no próximo capítulo...

## Sua caixa de ferramenta SQL

Dê uma salva de palmas para você.

Você já está mais que meio caminho andado no livro. Faça uma checagem em todos os termos-chave SQL que aprendeu no Capítulo 7. Para uma lista completa de dicas do livro, veja o Apêndice iii.

### Esquema

Uma descrição dos dados no seu banco de dados junto com qualquer outro objeto relacionado e a forma como eles se conectam.

### Relacionamento um-para-um

Exatamente uma linha da parent table é relacionada com uma linha da child table.

### Relacionamento um-para-muitos

Uma linha de uma tabela pode ter muitas linhas correspondentes em uma segunda tabela, mas a segunda tabela pode ter apenas uma linha correspondente na primeira.

### Relacionamento muitos-para-muitos

Dois tabelas estão conectadas por uma tabela de conexão permitindo muitas linhas na primeira a se corresponder muitas linhas na segunda, e vice-versa.

### Primeira Forma Normal (1NF)

Colunas contendo apenas dados atômicos e sem grupos repetidos de dados são permitidos em uma coluna.

### Dependência Funcional Transitória

Isto quer dizer que qualquer coluna não-chave está relacionada com qualquer uma das outras colunas não-chave.

### Segunda Forma Normal (2NF)

Sua tabela deve estar em 1NF e não conter nenhuma dependência funcional para ser 2NF.

### Terceira Forma Normal (3NF)

Sua tabela deve estar em 2NF e sem dependências transitórias.

### Chave externa

Sua tabela deve estar em 2NF e não ter nenhuma dependência transitória.

### Chave Composta

Esta é uma chave feita de múltiplas colunas, que cria um valor chave único.



## Aponte seu lápis

### Solução da página 234.

Use as funções ALTER e SUBSTRING\_INDEX para finalizar estas colunas. Escreva tantas consultas quanto forem necessárias.

**id contato**  
**sobrenome**  
**primeiro\_nome**  
**telefone**  
**email**  
**sexo**  
**aniversario**  
**profissao**  
**cidade**  
**estado**  
**estado\_civil**  
**interesse1**  
**interesse2**  
**interesse3**  
**interesse4**  
**procura**

Antes de tudo você precisa criar as novas colunas:

```
ALTER TABLE meus_contatos
ADD COLUMN interesse1 VARCHAR (50),
ADD COLUMN interesse2 VARCHAR (50),
ADD COLUMN interesse3 VARCHAR (50),
ADD COLUMN interesse4 VARCHAR (50);
```

Depois você vai precisar mover o primeiro interesse para a primeira coluna. Você pode fazer isto, desta forma:

```
UPDATE meus_contatos
SET interesse1 = SUBSTRING_INDEX (interesses, ',', 1);
```

Depois precisamos remover o primeiro interesse do campo interesses, já que ele foi armazenado na coluna interesse1. Nós removemos tudo até logo após a primeira vírgula com uma função string:

TRIM remove o espaço deixado na frente da linha de texto depois de termos removido em frente à vírgula.

```
UPDATE meus_contatos SET interesses = TRIM(RIGHT(interesses,
(LENGTH(interesses)-LENGTH(intere$$ - 1)));
```

RIGHT exibe parte da coluna interesses, começando do lado direito.

↑ Esta parte com aparência assustadora calcula quanto da coluna interesses nós precisamos. Ela pega o tamanho total da coluna interesses e subtrai o tamanho da parte que subtraímos para interesse1. Então subtraímos mais um, para começarmos depois da vírgula.

E agora repetimos aqueles passos para as outras colunas interesse:

```
UPDATE meus_contatos SET interesse2 = SUBSTRING_INDEX(interesses, ',', 1);
UPDATE meus_contatos SET interesses = TRIM(RIGHT(interesses, (LENGTH(interesses)-
LENGTH(intere$$ - 1)));
UPDATE meus_contatos SET interesse3 = SUBSTRING_INDEX(interesses, ',', 1);
UPDATE meus_contatos SET interesses = TRIM(RIGHT(interesses, (LENGTH(interesses)-
LENGTH(intere$$ - 1)));
```

Para a última coluna, tudo que nos resta ali é um só valor:

```
UPDATE meus_contatos SET interesse4 = interesses;
```

Agora podemos excluir a coluna interesses inteiramente. Também poderíamos ter renomeado-a para interesse1 e não precisaríamos do comando ADD COLUMN (supondo que tenhamos apenas quatro interesses).



Solução dos  
Exercícios da  
página 231.

Escreva uma consulta para Regis sem utilizar a coluna interesse.

```
SELECT * FROM meus_contatos  
WHERE sexo = 'F'  
AND estado_civil = 'solteira'  
AND estado = 'MA'  
AND procura LIKE '%homem solteiro%'  
AND aniversario > '1950-03-20'  
AND aniversario < '1960-03-20';
```

← Esta é essencialmente a mesma query que Greg usou para Nigel, exceto pelo fato de ter deixado de fora os juros.



## 8 Junções e operações multitabelas

### Não podemos todos nos entender?

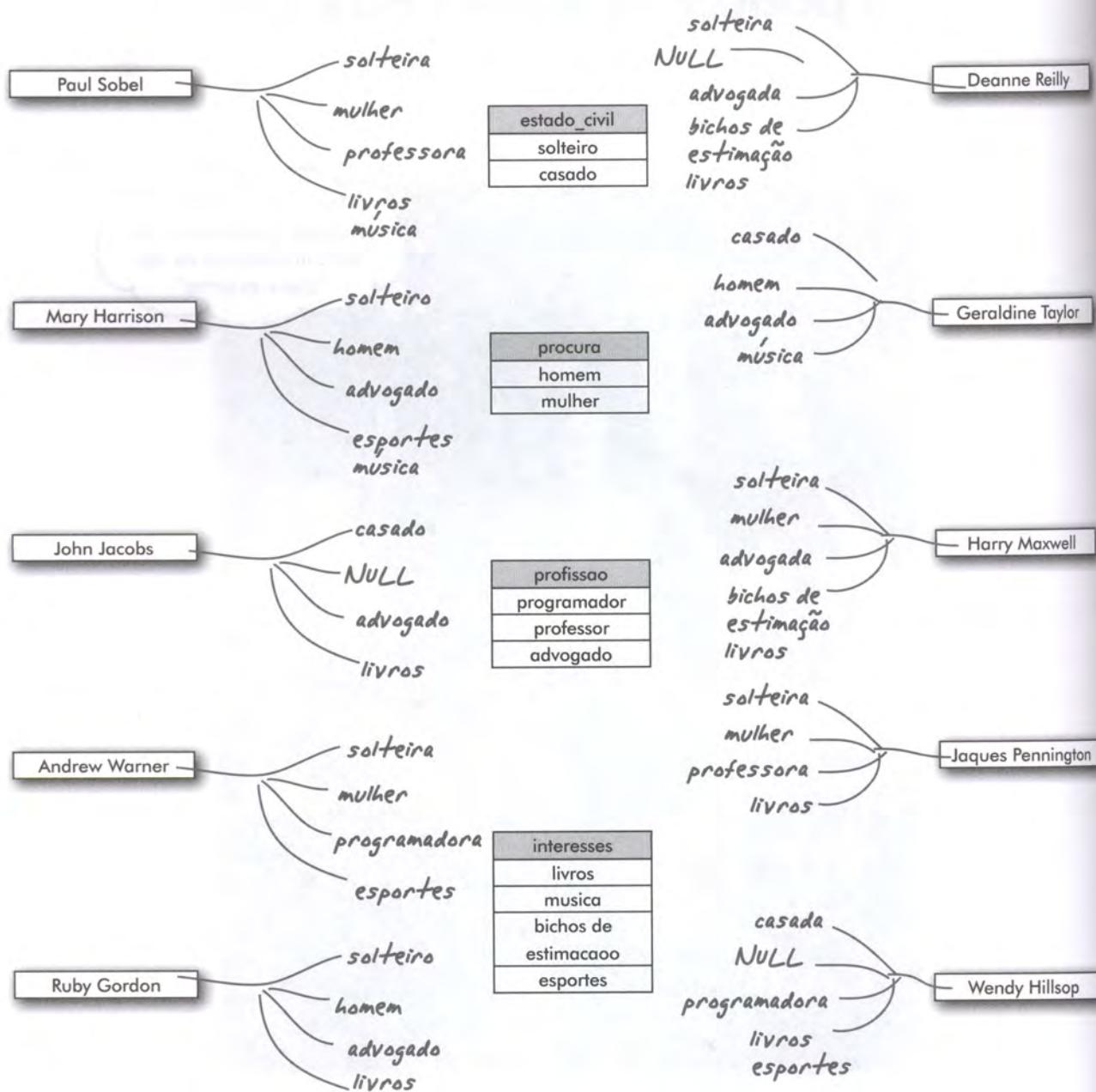


Por favor, vá embora  
Jacques. Nenhuma de nós  
está interessada na sua  
"chave externa".

**Bem-vindos a um mundo multitabela.** É ótimo ter mais de uma tabela em seu banco de dados, mas você terá algumas novas técnicas e ferramentas para trabalhar com elas. Com as múltiplas tabelas vem também confusão, então precisará de aliases (apelidos) para manter sua tabela correta. E conexões ajudam você a juntar tabelas, para que possa pegar todos os dados que espalhou. Prepare-se, é hora de tomar o controle do seu banco de dados novamente.

# Ainda repetindo, repetindo...

Greg percebeu os mesmos valores para **estado\_civil**, **profissao**, **interesses** e **procura** pipocando repetitivamente.



Pré-  
Ter mu  
povoar  
inter  
essas q  
tabela i  
Primei  
descob  
lista en

## Re-povoando suas tabelas

Muitos valores duplicados fará mais fácil para carregar as tabelas `estado_civil`, `profissao`, `interesses` e `procura`. Greg quer carregar essas quatro tabelas com os valores já inseridos na tabela `meus_contatos`.

Primeiro ele precisa consultar sua tabela para descobrir o que já está lá, mas ele não quer ter uma enorme de valores duplicados.

Não faria sentido ter um conjunto de listas de valores em algumas tabelas?

### Aponte seu lápis

Escreva consultas que possam recuperar os valores das colunas `estado_civil`, `profissão`, `interesses` e `procura` da antiga tabela `meus_contatos`, sem gerar qualquer duplicidade. Você pode querer consultar ao problema das vendas das Garotas Bandeirantes do Capítulo 6.



## Aponte seu lápis Solução

Escreva consultas que possam recuperar os valores das colunas estado\_civil, profissão, interesses e procura da antiga tabela meus\_contatos, sem gerar qualquer duplicidade. Você pode querer consultar ao problema das vendas das Garotas Bandeirantes do Capítulo 6.

```
SELECT FROM meus_contatos
GROUP BY estado_civil
ORDER BY estado_civil;
```

utilizar GROUP BY combina todos os valores duplicados em um só valor para cada grupo.

Então usando ORDER BY você tem uma lista alfabetica.

Se você não fizer nessa ordem, terá um erro. ORDER BY sempre precisa vir por último.

```
SELECT FROM meus_contatos
GROUP BY profissao
ORDER BY profissao;
```

```
SELECT FROM meus_contatos
GROUP BY procura
ORDER BY procura;
```

~~SELECT interesses  
FROM meus\_contatos  
GROUP BY interesses  
ORDER BY interesses;~~

Mas esta consulta não funciona para a coluna interesses. Nós temos múltiplos valores nesta coluna, lembra-se?



**Nós não podemos simplesmente utilizar um SELECT para remover os interesses para fora desta coluna**

Utilizar este comando SELECT para a coluna interesses não irá funcionar quando temos valores deste tipo:

Interesses
livros, esportes
música, bichos de estimação, livros
bichos de estimação, livros
esportes, música

## Nós tocamos o samba "esta tabela não é fácil de normalizar"

Como um cão que não tem nenhum osso, nosso nada normalizado projeto realmente nos prejudicou. Não há jeito fácil de tirar os valores para fora da uma interesses de uma forma que podemos vê-los um por vez.

Nós precisamos ir disto :

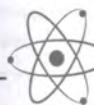
interesses
primeiro, segundo, terceiro, quarto

Nossa coluna da tabela meus\_contatos.

para isto

interesses
primeiro
segundo
terceiro
quarto

Uma coluna na nossa nova tabela interesses.



### PODER DO CÉREBRO

Como podemos pegar esses diversos valores e inseri-los em uma só coluna na tabela interesses?

Não podemos fazer isso manualmente? Quero dizer, eu posso olhar ao longo de cada linha de meus\_contatos e inserir cada valor para a nova tabela.



### Primeiro, isto daria um imenso trabalho. Imagine milhares de linhas

E fazer isso com as mãos seria ainda mais difícil de localizar uma duplicidade. Quando você tem centenas de interesses tem que olhar cada vez que inserir um novo interesse para ver se realmente ele estava lá.

Ao invés de fazer todo este trabalho árduo e arriscar uma porção de erros de digitação, vamos deixar o Sistema SQL fazer o trabalho tedioso para você.

## Nos interesses especiais (coluna)

O único e direto de fazer isso é adicionar quatro novas colunas para meus\_contatos onde podemos armazenar os valores à medida que nós preparamos e depois podemos eliminá-las quando tivermos terminado.

Aponte seu lápis

Você já sabe como utilizar o comando ALTER a esta altura, então você precisa de ALTER (alterar) meus\_contatos para ter quatro novas colunas. Nomeie-as interesse1, interesse2, interesse3, interesse4.

→ Respostas na página 306.

Aqui está a visual das colunas interesses e da nova coluna interesse na tabela meus\_contatos agora que você executou o ALTER.

Interesses	interesse1	interesse2	interesse3	interesse4
primeiro, segundo, terceiro, quarto				

Nós podemos facilmente copiar a primeira faixa de texto que corresponde ao primeiro interesse e colocá-la na coluna interesse1 com a nossa função SUBSTRING\_INDEX do Capítulo 5:

```
UPDATE meus_contatos
SET interesse1 = SUBSTRING_INDEX (interesses, ',', 1);
```

Execute o comando acima, e é isto que você terá:

Interesses	interesse1	interesse2	interesse3	interesse4
primeiro, segundo, terceiro, quarto	primeiro			

O nome da  
nossa coluna.  
O caractere a ser  
procurado: a vírgula.  
...procure  
pela primeira  
vírgula.

## Mantenha-se interessado

Agora para a parte complicada: iremos utilizar outra função substring\_index para remover da coluna interesses os dados que acabamos de mover para a coluna interesse1. Então poderemos preencher o restante das colunas interesse da mesma forma.

Interesses	interesse1	interesse2	interesse3	interesse4
primeiro, segundo, terceiro, quarto	primeiro			

Iremos remover o primeiro interesse, a vírgula que segue e o espaço depois da vírgula da coluna interesses

Iremos utilizar uma função SUBSTR que vai pegar a linha de texto na coluna interesses e retornar apenas com parte dela. Nós iremos dizer para retornar a mesma parte que colocamos na interesse1 e mais dois caracteres (para a vírgula e o espaço).

Tradução: Altere o valor na coluna interesses para ser o que quer que esteja lá agora, sem a parte colocada na coluna interesse1, sem a vírgula e o espaço depois dela.

O tamanho do texto no campo interesse1/...

...aumentando mais dois caracteres: um para a vírgula, outra para o espaço.

```
UPDATE meus_contatos
```

```
SET interesses = SUBSTR(intereses, LENGTH(interesse1)+2);
```

SUBSTR retorna parte da linha de texto original nesta coluna. Ela pega a linha e corta a primeira parte que especificamos entre os parênteses, e retorna a segunda parte.

Length retorna um número que é o tamanho de qualquer que seja a linha de texto constante nos parênteses que a segue.

Em nosso exemplo, o tamanho da linha first é de 5 caracteres.

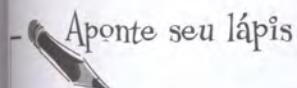
Ainda se lembra como algumas funções são diferentes dependendo da opção de SQL que você está usando? Bem, esta é uma delas. Refere-se a um guia de referências muito útil - como SQL em poucas palavras da O'Reilly - só que específico para seu SQL.

Então em nosso exemplo, o número retornado por LENGTH é 5+2 ou 7, que é o número de caracteres que serão removidos da primeira faixa na coluna interesses.

## UPDATE todos seus interesses

Depois de termos executado aquele comando UPDATE, nossa tabela está assim, mas ainda não acabamos. Nós temos que fazer a mesma coisa para as colunas interesse2, interesse3 e interesse4.

Interesses	interesse1	interesse2	interesse3	interesse4
primeiro, segundo, terceiro, quarto	primeiro			



Preencha as lacunas para completar o comando UPDATE de Greg.  
Nós lhe demos algumas observações para ajudar.

Dica: A coluna interesses será alterada a cada vez porque os valores da faixa de texto na coluna interesses estão sendo abreviados por uma função SUBSTR.

```
UPDATE meus_contatos SET
    interesse1 = SUBSTRING_INDEX(interesses, ',', 1),
    interesses = SUBSTR(interesses, LENGTH(interesse1)+2),
    interesse2 = SUBSTRING_INDEX(.....),
    interesses = SUBSTR(.....),
    interesse3 = SUBSTRING_INDEX(.....),
    interesses = SUBSTR(.....),
    interesse4 = .....
```

Depois de você ter removido os três primeiros interesses da coluna interesses, tudo que lhe resta é o quarto interesse. O que precisa ser feito aqui?

Preencha o que está em cada coluna depois deste comando enorme.

Interesses	interesse1	interesse2	interesse3	interesse4
primeiro, segundo, terceiro, quarto	primeiro			

## Pegando todos os interesses

Conseguimos separar todos os interesses finalmente. Nós podemos chegar até eles com simples comandos SELECT, mas podemos acessá-los todos de uma só vez, e podemos facilmente removê-los em um só resultado conjunto, uma vez que eles estão em quatro colunas. Quando tentamos, conseguimos o seguinte resultado:

File Edit Window Help TooManyColumns			
interesse1	interesse2	interesse3	interesse4
primeiro	segundo	terceiro	quarto
cavalos	animais de estimação	livros	filmes
música	pescar		
pintar			
cavalos	animais de estimação	livros	velejar
música	esportes		
viajar	música		
cavalos	animais de estimação	livros	tricô
música	esportes	viajar	
animais de estimação	escrever		
cachorros	trilha		
filmes	esportes		

Mas pelo menos podemos escrever quatro comandos SELECT separados para tirar todos os valores de lá:

```
SELECT interesse1 FROM meus_contatos;    SELECT interesse3 FROM meus_contatos;
SELECT interesse2 FROM meus_contatos;    SELECT interesse4 FROM meus_contatos;
```

Tudo o que nos falta agora é uma maneira de retirar aqueles comandos SELECT e encher nossas novas tabelas com esse conteúdo. Não há só um jeito de fazer isso; há pelo menos três!



Exercícios

### Tente em casa

Considere o comando SELECT que você escreveu para a coluna profissão na página 345:

```
SELECT profissao FROM meus_contatos GROUP BY profissao
ORDER BY profissao;
```

Na próxima página nós vamos lhe mostrar TRÊS MANEIRAS de se aproveitar destes comandos SELECT para conseguir que sua nova tabela interesses esteja pré-povoada.

Brinque um pouco com SELECT, INSERT e CREATE para ver o que consegue inventar e então olhe para na próxima página para ver as três maneiras.

O objetivo aqui não é conseguir acertar o exercício, mas pensar nas suas possibilidades.

## Muitos caminhos para um só lugar

quanto estiver apto a fazer a mesma coisa: três ou mais jeitos diferentes pode ser divertido para os palhaços loucos, pode ser confuso para o resto de nós.

Entretanto é útil. Quando você sabe três maneiras de se fazer algo, pode-se saber o jeito que melhor se encaixa nas suas necessidades. E à medida que os dados crescem, você perceberá que algumas consultas são realizadas mais rapidamente pelo seu Sistema SQL. Quando suas tabelas crescem muito, irá querer otimizar suas consultas, então saber que você pode fazer esta mesma tarefa em diferentes formas vai lhe ajudar a fazer isto:

Nas próximas páginas estão todas as três maneiras que você pode criar e popular esta tabela com valores distintos e ordenados alfabeticamente.

profissao
id_prof
profissão



Vocês sabem o que é divertido em relação ao SQL, crianças.

Geralmente existe mais de um jeito de fazer alguma coisa.



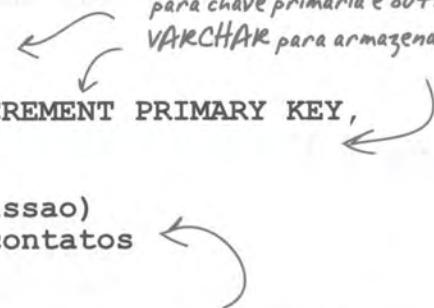
## CREATE, SELECT, e INSERT a (quase) o mesmo tempo

### CREATE TABLE, daí INSERT com SELECT

Você sabe como fazer essa! Primeiro cria a tabela profissao, então vai encher as colunas com os valores obtidos como o comando SELECT da página 345.

```
CREATE TABLE profissao
(
    id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    profissão varchar(20)
);
INSERT INTO profissao (profissao)
SELECT profissao FROM meus_contatos
GROUP BY profissao
ORDER BY profissao;
```

Crie a tabela profissão com uma coluna para chave primária e outra coluna do tipo VARCHAR para armazenar as profissões.



Agora preencha a coluna profissão da tabela profissão com os valores obtidos com os valores do seu comando SELECT

### CREATE TABLE com SELECT, daí use o ALTER para adicionar a chave primária

Segunda maneira: CREATE a tabela profissao usando os dados de um comando SELECT capaz de obter todos os dados da coluna profissão da tabela meus\_contatos, daí então utilize o ALTER e adicione um campo de chave primária.

```
CREATE TABLE profissão AS
SELECT profissao FROM meus_contatos
GROUP BY profissao
ORDER BY profissao;
ALTER TABLE profissao
ADD COLUMN id INT NOT NULL AUTO_INCREMENT FIRST,
ADD PRIMARY KEY (id);
```

Cria a tabela profissão com uma coluna cheia com os valores do comando SELECT...

...e, então, utilize o ALTER para adicionar à tabela um campo de chave primária.

# CREATE, SELECT e INSERT ao mesmo tempo

## 3. CREATE TABLE com uma chave primária e com o SELECT tudo de uma vez

Este é o jeito de um só passo: Crie uma tabela profissao com uma coluna para chave primária e uma coluna VARCHAR para armazenar os valores profissão e, ao mesmo tempo, encha-a com os valores originados pelo comando SELECT. SQL auto-incrementa para que seu Sistema SQL saiba que a coluna id deverá ser alimentada automaticamente, o que nos deixa apenas uma coluna, que é para onde vão os dados.

```
CREATE TABLE profissao
(
    id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    profissao varchar(20)
) AS
SELECT profissao FROM meus_contatos
GROUP BY profissao
ORDER BY profissao;
```

Cria uma tabela profissão  
já com ambas as colunas de  
chave primária e profissão,  
preenche a coluna profissão  
com os valores do SELECT.

Eu nunca vi AS antes. Parece que  
ele foi utilizado para fazer referência dos  
resultados de uma consulta para serem  
inseridos em uma nova tabela.



### Sim. A palavra-chave AS faz exatamente isso

É tudo uma questão de usar, aliás, é sobre o que falaremos logo!

## O que há com este AS?

AS povou a nova tabela com o resultado do SELECT. Então quando utilizarmos no segundo e no terceiro exemplo, nós estávamos dizendo ao programa para pegar todos os valores que saíram da tabela meus\_contatos como um resultado do SELECT e colocá-los na nova tabela profissao que recém-criamos.

Se não tivesse especificado que a nova tabela tem duas colunas com novos nomes, AS teria criado apenas uma coluna, preenchida com os mesmos nomes e tipos de dados das colunas resultantes do SELECT.

Nós estávamos criando  
uma coluna VARCHAR  
na nossa nova tabela e  
chamando-a de profissão.

Esta pequena palavra-chave  
está fazendo grandes coisas.  
Elas estão afunilando todos os  
dados de saída para uma nova  
tabela.

```
CREATE TABLE profissao
(
    id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    profissao varchar(20)
) AS
SELECT profissao FROM meus_contatos
GROUP BY profissao
ORDER BY profissao;
```

Se não tivessemos criado duas  
colunas para a nova tabela, AS teria  
criado apenas uma tabela e preenchido  
com os mesmos nomes e tipos de  
dados que a coluna resultante do  
SELECT.

Todos esses se referem à coluna  
profissão em meus\_contatos  
porque todos eles são partes  
do SELECT.

que criamos a tabela profissão com uma chave-primária auto-incrementadora, nós precisávamos adicionar valores para a segunda coluna na tabela, a qual nomeamos de profissao.



Estou confusa, "profissão" aparece cinco vezes naquela única consulta. O Sistema SQL deve saber qual é qual, mas como eu poderia saber?

### **SQL permite que você determine um apelido para um nome de coluna para que você não fique confuso**

É por isso que o Sistema SQL permite que você dê novos nomes temporariamente para suas colunas e tabelas, conhecidas como aliases (pseudônimos).

## Pseudônimos de coluna

Um pseudônimo não poderia ser mais fácil. Nós o colocaremos logo após a inicial do nome da coluna na sua consulta com outro AS para dizer ao Sistema para referir-se à coluna profissao em meus\_contatos como outro novo nome que deixa mais claro o que está acontecendo.

Chamaremos os valores da profissão da qual estamos selecionando da tabela meus\_contatos de **mc\_prof** (mc é abreviação para meus\_contatos).

```
CREATE TABLE profissao
(
    id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    profissao varchar(20)
) AS
SELECT profissao AS mc_prof FROM meus_contatos
GROUP BY mc_prof
ORDER BY mc_prof;
```

Esta consulta faz exatamente a mesma coisa que a primeira, mas graças ao pseudônimo ela é mais fácil de se compreender.

Coloque o pseudônimo logo após o primeiro uso do nome original da coluna na consulta para dizer ao Sistema para se referir a ela como o pseudônimo, de agora em diante.

Há uma pequena diferença entre as duas consultas. Todas as consultas retornam os resultados em forma de tabelas. **O pseudônimo altera o nome da coluna nos resultados, mas não altera o nome original da coluna em nenhuma forma.** O pseudônimo é temporário.

Isso já que excedemos os resultados ao especificar que nossa nova tabela possui duas colunas - a chave primária e a coluna profissão - nossa nova tabela ainda terá uma coluna chamada profissao e não mc\_prof.

profissao
programador
professor
advogado

A consulta original resulta no nome original da coluna.

Os resultados da consulta utilizando o pseudônimo. O nome da coluna é o mesmo que o pseudônimo.

mc_prof
programador
professor
advogado

## Pseudônimos de tabelas, quem precisa deles?

Você! Nós estamos prestes a mergulhar o Use a Cabeça no mundo de conexões, onde iremos selecionar dados de mais de uma tabela e sem pseudônimos. Você ficará cansado de escrever o nome daquelas tabelas repetidamente.

Crie pseudônimos para sua tabela da mesma forma que cria pseudônimos para colunas. Coloque o pseudônimo da tabela depois do primeiro uso do nome da tabela na consulta com outro AS para dizer ao seu Sistema para se referir à tabela original `meus_contatos` como mc, por ora.

```
SELECT profissao AS mcprof
FROM meus_contatos AS mc
GROUP BY mc_prof
ORDER BY mc_prof;
```

*Crie seu pseudônimo para tabela exatamente da mesma forma que você faz para criar pseudônimo para colunas.*

Pseudônimos para tabelas também são chamados de nomes correlacionais.

*Não há diferença no que estas duas tabelas fazem.*

*Tenho que utilizar "AS" todas as vezes que eu for definir um pseudônimo?*

**Não, há um atalho para definir seus pseudônimos.**

Apenas omita a palavra AS. A consulta abaixo faz exatamente a mesma coisa que a consulta no topo da página.

```
SELECT profissao mcprof
FROM meus_contatos mc
GROUP BY mc_prof
ORDER BY mc_prof;
```

*Removemos o AS. Isto funciona enquanto o pseudônimo seguir logo após o nome da coluna ou tabela, e ele irá funcionar.*



## Tudo o que você gostaria de saber sobre conexões internas

Você que já ouviu qualquer pessoa conversando sobre SQL, provavelmente ouviu a palavra "join" (conexão) atirada ao ar. Elas não são tão complicadas como deve achar que elas são. Nós iremos apresentá-las, mostrar como elas funcionam, e lhe dar chances o suficiente quando se deve usar conexões. E qual conexão deverá utilizar.

Mas antes de irmos ao assunto, vamos começar com o tipo mais simples de conexão (ela não é verdadeiramente uma conexão).

Ela tem diversos nomes. Nós a chamaremos neste livro de **conexão cartesiana**, mas também é chamada de produto cartesiano, produto cruzado, conexão cruzada, e também estranhamente chamada de "sem conexão".

Suponha que você tem uma tabela com nome de crianças e outra tabela com os brinquedos que estas crianças possuem. Está por sua conta saber que brinquedo comprar para qual criança.

...e foi daí que tabelas com pequenos resultados se originaram.

brinquedos

id_brinquedo	brinquedo
1	bambolê
2	planador
3	soldados
4	gaita
5	figurinhas

garotos

id_garoto	garoto
1	Davey
2	Bobby
3	Beaver
4	Richie



## Conexão cartesiana

A consulta abaixo nos dá os resultados cartesianos quando consultamos ambas as tabelas de uma só vez pela coluna brinquedo da tabela brinquedos e pela coluna garoto da tabela garotos.

```
SELECT b.brinquedo, g.garoto
FROM brinquedos AS b
CROSS JOIN
garotos AS g;
```

Estamos usando pseudônimos de tabelas aqui também.

Lembra-se das nossas anotações do último capítulo? O nome antes do ponto é a tabela, e o nome depois do ponto é o nome de uma coluna naquela tabela. Apenas agora estamos usando pseudônimos de tabelas ao invés do nome completo.

Esta linha diz SELECT (seleciona) a coluna chamada garoto da tabela garotos e a coluna chamada brinquedo da tabela brinquedos. E o restante da consulta conecta aquelas duas colunas em uma nova tabela de resultados.

A conexão Cartesiana pega cada valor da primeira tabela e compara com cada valor da segunda tabela.

id_brinquedo.brinquedo	id_garoto.garoto
bambolê	Davey
planador	Bobby
soldados	Beaver
gaita	Richie
figurinhas	

A conexão cruzada resulta em cada linha de uma tabela cruzado com cada linha da segunda.

Estas linhas mostram o resultado da conexão. Cada brinquedo é conectado a cada garoto. Não há duplicidade.

Esta conexão nos dá 20 resultados. Isto é, 5 brinquedos x 4 garotos para contabilizar cada combinação possível.

Só porque brinquedos.brinquedo tinha mais resultados é que ela aparece primeiro nos resultados. Se tivéssemos 5 resultados para a coluna garoto e 4 para brinquedos, você veria a coluna garotos agrupada primeiro. Mas lembre-se, a ordem dos resultados não tem nenhum significado com esta consulta.

brinquedo	garoto
bambolê	Davey
bambolê	Bobby
bambolê	Beaver
bambolê	Richie
planador	Davey
planador	Bobby
planador	Beaver
planador	Richie
soldados	Davey
soldados	Bobby

## não existem Perguntas Idiotas

P: Quando é que eu usaria isto?

R: É importante saber sobre isso porque quando você estiver manipulando com conexões, poderá accidentalmente obter resultados cartesianos. Isto lhe ajudará a pensar em como consertar sua conexão. Isto realmente pode acontecer algumas vezes. Às vezes, conexões cruzadas são utilizadas para testar a velocidade e a configuração do seu Sistema SQL. O tempo que elas demoram é mais fácil para detectar e comparar quando utilizar uma consulta devagar.

P: Digamos que eu use a consulta dele:

`SELECT * FROM brinquedos CROSS JOIN garotos;`  
O que acontece se eu utilizar `SELECT *?`

R: Tente você mesmo, mas ainda acabaria com 20 linhas; eles apenas incluiriam todas as colunas.

Uma conexão interna é uma conexão cruzada com alguns resultados removidos por uma condição na consulta.

P: E seu eu fizer uma conexão cruzada entre duas tabelas bem grandes?

R: Você vai obter uma quantidade enorme de resultados. É melhor não fazer este tipo de conexão, você corre o risco de travar sua máquina devido à quantidade de dados que o Sistema tem que retornar.

P: Existe outra sintaxe para esta consulta?

R: Pode apostar que sim. Você pode tirar as palavras CROSS JOIN e usar apenas uma vírgula em seu lugar, desta forma:  
`SELECT brinquedos.brinquedo, garotos.garoto FROM  
brinquedos, garotos;`

P: Já ouvi o termo "inner join"(conexão interna) e "outer join" (conexão externa) sendo usado. Esta conexão cartesiana é a mesma coisa?

A conexão cartesiana é um tipo de conexão interna. Uma conexão interna é basicamente apenas uma conexão cartesiana onde algumas linhas do resultado são removidas por uma condição na consulta. Iremos olhar conexões internas nas próximas páginas, então guarde esta pergunta!

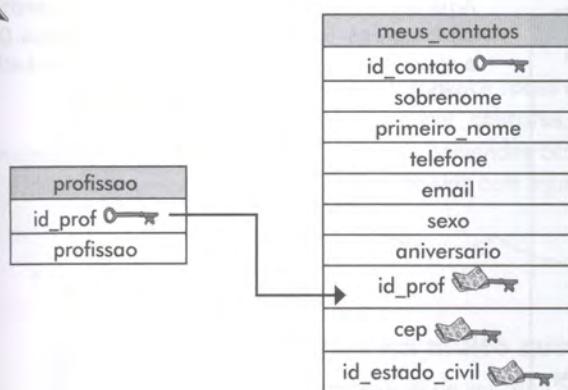


### Musculação cerebral

O que você acha que seria o resultado desta consulta?

```
SELECT g1.garoto, g2.garoto
FROM garotos AS b1 CROSS JOIN garotos AS b2;
Tente você mesmo.
```

Aponte seu lápis



Aqui estão duas tabelas da estrutura do banco de dados gregs\_list: profissao e meus\_contatos. Observe a consulta e escreva nas lacunas o que você acha que cada linha da consulta está fazendo.

```

SELECT mc.sobrenome,
       mc.primeiro_nome,
       p.profissao
  FROM meus_contatos AS mc
 INNER JOIN
       profissao AS p
  ON mc.id_contato = p.id_prof;
  
```

Suponha que os dados nos adesivos abaixo estão nas tabelas. Desenhe como a tabela resultado ficará com os resultados.

Joan Everett  
solteiro  
4-3-1978  
Salt Lake City, UT  
Artista  
Feminina  
jeverett@mightygumball.net  
velejar, trilhar, cozinhar  
555 555-9870

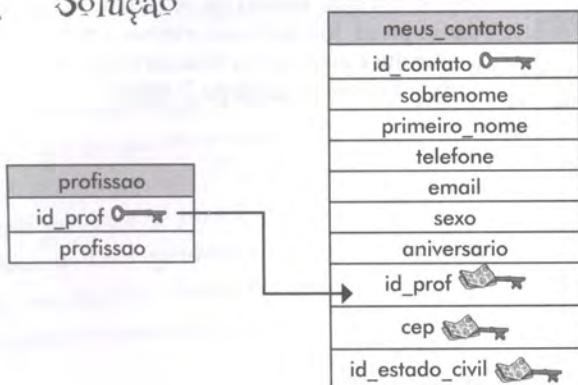
Paul Singh  
Casado  
12-10-1980  
New York City, NY  
Professor  
masculino  
ps@tikibeanlounge.com  
cachorros, espeluncologia  
555 555-8222

Tara Baldwin  
Casada  
9-1-1970  
Boston, MA  
Chefe  
Feminina  
tara@breakneckpizza.com  
filmes, ler, cozinhar  
555 555-3432



## Aponte seu lápis

### Solução



```

SELECT mc.sobrenome,
       mc.primeiro_nome,
       p.profissao
  FROM meus_contatos AS mc
 INNER JOIN
       profissao AS p
     ON mc.id_contato = p.id_prof;
    
```

Selecionar a coluna sobrenome na tabela meus contatos  
(pseudônimo mc)

e a coluna primeiro\_nome na tabela meus\_contatos

e a coluna profissao na tabela profissao (pseudônimo p)

a partir da tabela meus\_contatos e

utilize uma conexão interna para conectar os resultados  
do SELECT com

a tabela profissao (pseudônimo p)

Suponha que os dados nos adesivos abaixo estão nas tabelas. Desenhe  
como a tabela resultado ficará com os resultados.

sobrenome	primeiro_nome	profissao
Everett	Joan	artista
Singh	Paul	professor
Baldwin	Tara	chefe

## Liberando sua conexão interna



Entendi! É assim que eu conecto todas as minhas novas tabelas à nova tabela meus\_contatos. Não tenho que escrever uma porção de comandos SELECT, eu só preciso conectar minhas tabelas com aquela INNER JOIN (conexão interna) e estarei pronto!

### Ainda há muito que aprender!

Você acabou de ver uma pequena variação de um tipo de conexão. E você ainda tem muito que aprender sobre esta conexão e sobre outras conexões antes de estar apto a utilizá-las apropriadamente e de forma efetiva.

Uma CONEXÃO INTERNA combina os registros de duas tabelas usando operadores de comparação em uma condição. Colunas são retornadas apenas onde as linhas conectadas combinam com a condição. Vamos olhar de perto a sintaxe.

A palavra-chave WHERE também funciona aqui.

*Qualquer que seja a coluna que você precisa ver.*

**SELECT algumacoluna**

**FROM tabel1** ← *Não utilizamos os pseudônimos para simplificar o exemplo.*

**INNER JOIN** ←

**tabela2** ←

→ **ON alguma condição** ← *Esta condição pode utilizar qualquer um dos operadores de comparação.*

Uma CONEXÃO INTERNA combina os registros de duas tabelas usando operadores de comparação em uma condição.

### conexão interna em ação: a equijoin

Em consideração estas duas tabelas. Cada garoto tem apenas um brinquedo. Nós temos um relacionamento um-a-um, e id\_brinquedo é a chave externa.

garotos

id_garoto	garoto	id_brinquedo
1	Davey	3
2	Bobby	5
3	Beaver	2
4	Richie	1

brinquedos

id_brinquedo	brinquedo
1	bambolê
2	planador
3	soldados
4	gaita
5	figurinhas

Tudo que queremos fazer é descobrir que brinquedo cada garoto tem. Nós podemos utilizar nossa conexão interna com o sinal de = para combinar a chave externa na tabela garotos com a chave primária na tabela brinquedos e verificar qual brinquedo é indicado.

Conexões internas EQUÍJOIN testam a igualdade

```
SELECT garotos.garoto, brinquedos.brinquedo
FROM garotos
INNER JOIN
brinquedos
ON garotos.id_brinquedo = brinquedos.id_brinquedo;
```

id_garoto	garoto	id_brinquedo
1	Davey	3
2	Bobby	5
3	Beaver	2
4	Richie	1

id_brinquedo	brinquedo
1	bambolê
2	planador
3	soldados
4	gaita
5	figurinhas

Nossa tabela resultado.  
Poderíamos ter adicionado  
ORDER BY garotos.garoto se  
quiséssemos.

garoto	brinquedo
Richie	bambolê
Beaver	planador
Davey	soldados
Bobby	figurinhas

 Aponte seu lápis

Escreva as consultas equijoin para o banco de dados gregs\_list abaixo.

Consulta que retorne o email e a profissão de cada pessoa em meus\_contatos.

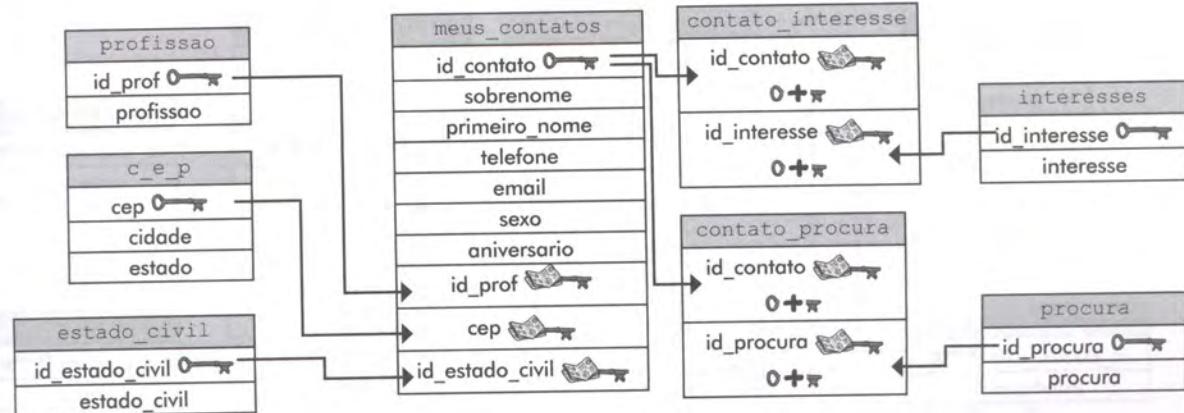
.....

Consulta que retorne o primeiro nome, sobrenome e o estado civil de cada pessoa em meus\_contatos.

.....

Consulta que retorne o primeiro nome, sobrenome e o estado de cada pessoa em meus\_contatos.

.....





## Aponte seu lápis Solução

Escreva as consultas equijoin para o banco de dados gregs\_list abaixo.

Consulta que retorne o email e a profissão de cada pessoa em meus\_contatos.

```
SELECT mc.email, p.profissao FROM meus_contatos mc
INNER JOIN profissao p ON mc.id_prof = p.id_prof;
```

← A chave externa id\_prof conecta com id\_prof na tabela profissão.

Consulta que retorne o primeiro nome, sobrenome e o estado civil de cada pessoa em meus\_contatos.

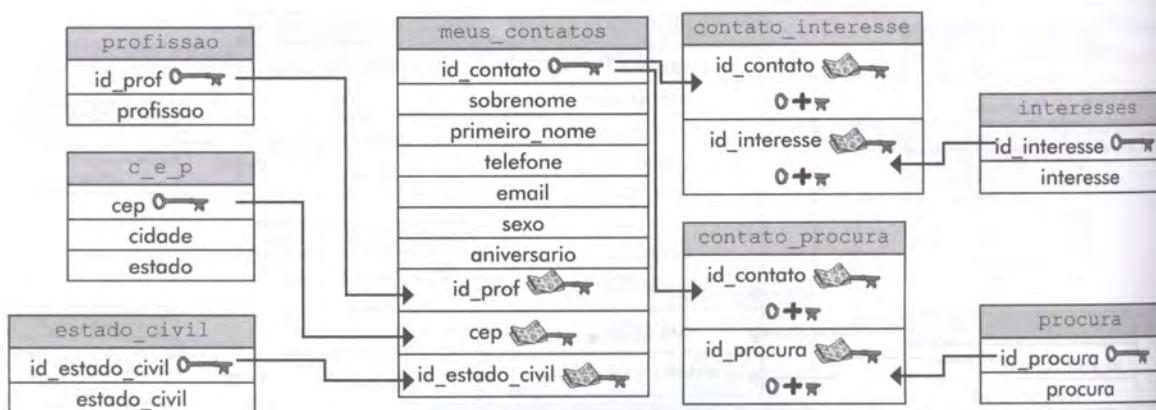
```
SELECT mc.primeiro_nome, mc.sobrenome, ec.estado_civil FROM meus_contatos mc
INNER JOIN estado_civil ec ON mc.id_estado_civil = ec.id_estado_civil;
```

← A chave externa id\_estado\_civil conecta com id\_estado\_civil na tabela estado\_civil.

Consulta que retorne o primeiro nome, sobrenome e o estado de cada pessoa em meus\_contatos.

```
SELECT mc.primeiro_nome, mc.sobrenome, c.estado FROM meus_contatos mc
INNER JOIN c_e_p c ON
mc.cep = c.cep;
```

← Desta vez estamos utilizando cep como chave que conecta as duas tabelas.



## conexão interna em ação: a não-equijoin

**não-equijoin** retorna qualquer linha que não for igual. Considere as mesmas duas tabelas, garotos e brinquedos. Ao usar a não-equijoin, podemos exatamente quais brinquedos cada garoto **não** tem (o que poderia ser útil se o garoto está próximo dos seus aniversários).

```
SELECT garotos.garoto, brinquedos.brinquedo
FROM garotos
INNER JOIN
    brinquedos
ON garotos.id_brinquedo <> brinquedos.id_brinquedo
ORDER BY brinquedos.brinquedo;
```

*(Viver nossos resultados facilitará a leitura.)*

garotos		
<i>id_garoto</i> 0 →	garoto	<i>id_brinquedo</i> ↗
1	Davey	3
2	Bobby	5
3	Beaver	2
4	Richie	1

toys	
<i>id_brinquedo</i> 0 →	brinquedo
1	bambolê
2	planador
3	soldados
4	harmonica
5	baseball cards

garoto	brinquedo
Beaver	bambolê
Beaver	soldados
Beaver	gaita
Beaver	figurinhas
Bobby	soldados
Bobby	gaita
Bobby	bambolê
Bobby	planador
Davey	bambolê
Davey	planador
Davey	gaita
Davey	figurinhas
Richie	planador
Richie	soldados
Richie	gaita
Richie	figurinhas

*Estes são os quatro brinquedos que Beaver ainda não tem.*

**NÃO-EQUIJOIN**  
Conexões internas  
testam por  
desigualdade.

## A última conexão interna: a conexão natural

Resta apenas um tipo de conexão interna e ela é chamada de **conexão natural**. Conexões naturais só funcionam se a **coluna a qual você está conectando possuir o mesmo nome nas duas tabelas**. Leve em consideração estas duas tabelas novamente:

garotos			brinquedos		
id_garoto	garoto	id_brinquedo	id_brinquedo	brinquedo	
1	Davey	3		bambolê	
2	Bobby	5		planador	
3	Beaver	2		soldados	
4	Richie	1		gaita	
				figurinhas	

Como anteriormente, nós queremos saber qual brinquedo cada criança tem. Nossa conexão natural irá reconhecer a coluna com nome igual em cada tabela e retornar a linha que se ajusta.

```
SELECT garotos.garoto, brinquedos.brinquedo
FROM garotos
NATURAL JOIN
brinquedos;
```

id_garoto	garoto	id_brinquedo	brinquedo
1	Davey	3	bambolê
2	Bobby	5	planador
3	Beaver	2	soldados
4	Richie	1	gaita
			figurinhas

Obtivemos exatamente o mesmo resultado que tivemos com nossa primeira conexão interna: a equijoin.

garoto	brinquedo
Richie	bambolê
Beaver	planador
Davey	soldados
Bobby	figurinhas

**CONEXÃO NATURAL**  
Conexões internas  
identificam colunas com  
nomes iguais.

 Aponte seu lápis

Escreva as consultas para o banco de dados gregs\_list  
abaixo como conexões naturais ou não-equi joins:

Consulta que retorne o email e a profissão de cada pessoa em meus\_contatos.

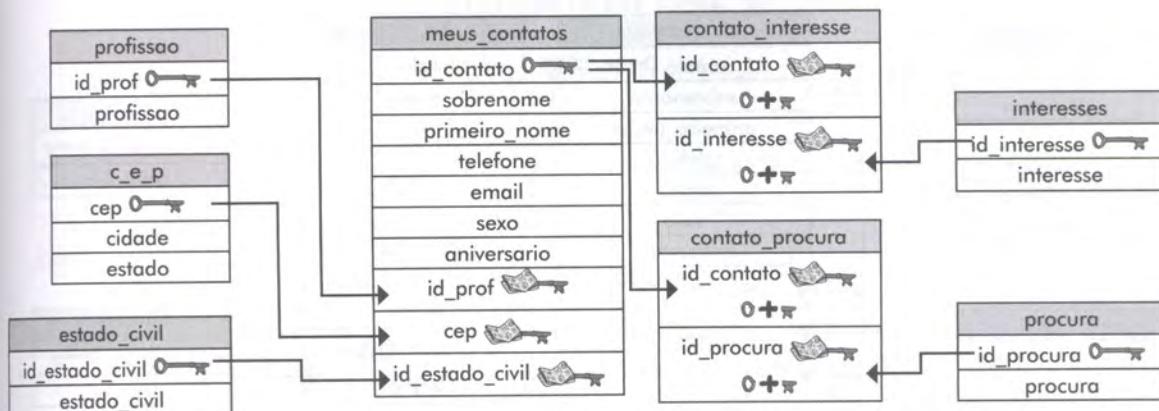
.....  
.....  
.....

Consulta que retorne o primeiro nome, sobrenome e o estado civil que a pessoa não é em meus\_contatos.

.....  
.....  
.....

Consulta que retorne o primeiro nome, sobrenome e o estado de cada pessoa em meus\_contatos.

.....  
.....  
.....





## Aponte seu lápis

### Solução

Escreva as consultas para o banco de dados gregs\_list  
abaixo como conexões naturais ou não-equi joins:

Consulta que retorne o email e a profissão de cada pessoa em meus\_contatos.

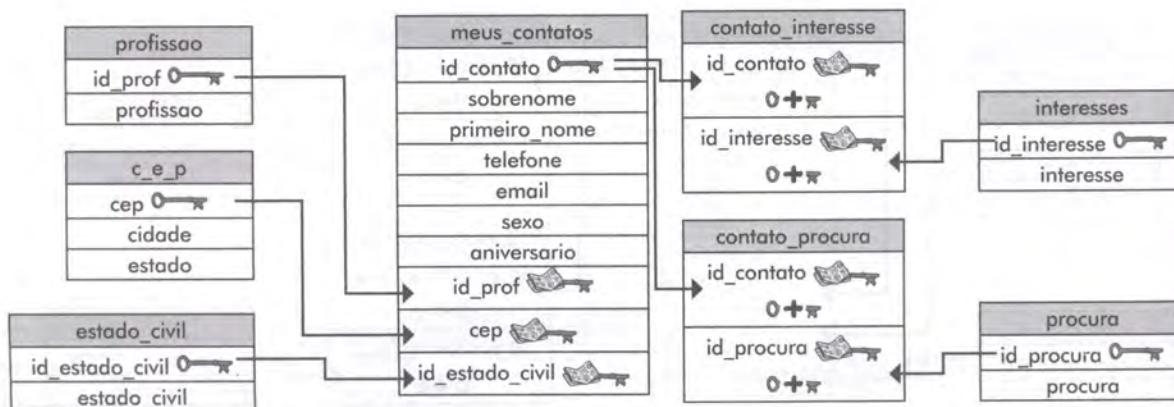
```
SELECT mc.primeiro_nome, mc.sobrenome, ec.estado_civil FROM meus_contatos mc INNER JOIN estado_civil ec ON mc.id_estado_civil <> ec.id_estado_civil;
```

Consulta que retorne o primeiro nome, sobrenome e o estado civil que a pessoa não é em meus\_contatos.

```
SELECT mc.primeiro_nome, mc.sobrenome, ec.estado_civil FROM meus_contatos mc INNER JOIN estado_civil ec ON mc.id_estado_civil <> ec.id_estado_civil;
```

Consulta que retorne o primeiro nome, sobrenome e o estado de cada pessoa em meus\_contatos.

```
SELECT mc.primeiro_nome, mc.sobrenome, c.estado FROM meus_contatos mc INNER JOIN c_e_p c;
```



## QUEM FAZ O QUÊ?

Associe cada junção com a descrição do que ela faz. Mais de uma junção pode estar associada a uma descrição

Conexão natural

Eu retorno todas as linhas onde uma coluna de uma tabela não combina com a coluna da outra tabela.

Equijoin

A ordem com que você conecta as tabelas importa para mim.

Conexão cruzada

Eu retorno todas as linhas onde uma coluna de uma tabela combina com a coluna da outra tabela, e utilizo a palavra-chave ON.

Não-equijoin

Eu combino duas tabelas que possuem o mesmo nome.

Junção interna

Eu posso retornar linhas iguais ao produto da linha de duas tabelas

Conexão cartesiana

Eu retorno todas as linhas possíveis e não possuo nenhuma condição.

Produto cruzado

Eu combino duas tabelas com uma condição.

## SOLUÇÃO DO QUEM FAZ O QUÊ?

Associe cada junção com a descrição do que ela faz. Mais de uma junção pode estar associada a uma descrição

Conexão natural

Eu retorno todas as linhas onde uma coluna de uma tabela não combina com a coluna da outra tabela.

Equijoin

A ordem com que você conecta as tabelas importa para mim.

Isto ainda está por vir no capítulo 10.

Conexão cruzada

Eu retorno todas as linhas onde uma coluna de uma tabela combina com a coluna da outra tabela, e utilizo a palavra-chave ON.

Não-equijoin

Eu combino duas tabelas que possuem o mesmo nome.

Junção interna

Eu posso retornar linhas iguais ao produto da linha de duas tabelas

Conexão cartesiana

Eu retorno todas as linhas possíveis e não possuo nenhuma condição.

Produto cruzado

Eu combino duas tabelas com uma condição.



## Exercícios

Utilize o diagrama do banco de dados gregs\_list abaixo para escrever consultas SQL para obter a informação requerida.

Escreva duas consultas, cada uma com uma conexão diferente para obter os dados que se ajustam às tabelas meus\_contatos e contato\_interesse.

```
SELECT mc.primeiro_nome, mc.sobrenome, ci.id_interesse FROM meus_contatos mc
INNER JOIN contato_interesse ci ON mc.id_contato = ci.id_contato;
```

```
SELECT mc.primeiro_nome, mc.sobrenome, ci.id_interesse FROM meus_contatos mc
NATURAL JOIN contato_interesse ci;
```

Escreva uma consulta que retorne todas as combinações possíveis de linhas de contato\_procura e procura.

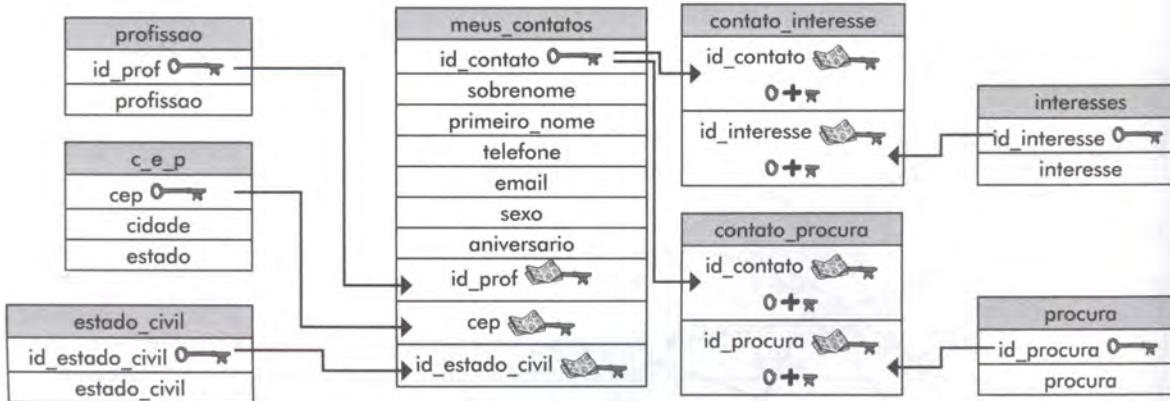
```
SELECT * FROM contato_procura CROSS JOIN procura;
```

```
SELECT * FROM contato_procura, procura;
```

Há duas formas de fazer a mesma conexão cruzada.

Liste as profissões das pessoas de meus\_contatos, mas sem duplicidade e em ordem alfabética.

```
SELECT p.profissao FROM meus_contatos mc
INNER JOIN profissao p ON mc.id_prof = p.id_prof GROUP BY profissao ORDER BY profissao;
```



## não existem Perguntas Ídiotas

**P:** Você pode conectar mais de duas tabelas?

**R:** Você pode, e falaremos mais sobre isso posteriormente. Agora, nosso foco é assimilar os conceitos das conexões.

**P:** Conexões não deveriam ser mais complicadas do que isso?

**R:** Depois que você começa a lidar com conexões e pseudônimos, consultas SQL parecem menos com uma língua inglesa e parece mais uma língua estrangeira. Também ao utilizar abreviações (como repor as palavras INNER JOIN por vírgulas nas consultas, por exemplo) poderia tornar as coisas ainda mais confusas. Por esta razão, este livro favorece mais a SQL detalhada ao invés da reduzida por diversas abreviações.

**P:** Isto quer dizer que existem outros jeitos de escrever consultas de conexões internas?

**R:** Sim, mas se você entendê-las através da sintaxe que apresentamos, recolhendo sintaxe de outros será fácil. Os conceitos são muito mais importantes do que usar WHERE ou JOIN em uma conexão.

**P:** Eu percebi que você utilizou ORDER BY em uma conexão. Isto quer dizer que vale tudo aqui também?

**R:** Yes. Sinta-se à vontade para utilizar GROUP BY e cláusulas WHERE, bem como funções com SUM e AVG a qualquer tempo.

## Consultas conectadas?

Greg está começando a gostar das conexões. Ele está começando a enxergar que ter tabelas múltiplas faz sentido, e elas não são difíceis de se trabalhar, se bem projetadas. Ele já até tem alguns planos para expandir a gregs\_list.



Mas ainda me vejo digitando uma consulta, ao invés de utilizar aqueles resultados em uma segunda consulta quando já me parece que eu seria capaz de fazer tudo em uma só... Não seria ótimo se eu pudesse colocar uma consulta dentro de outra consulta?

Mas isso é loucura.

**Uma consulta dentro de outra consulta?  
Isto é possível?**



## Pseudônimos de Colunas e Tabelas Revelados

A entrevista desta semana:  
Do que você está se escondendo?

**Use a Cabeça:** Bem-vindo, Pseudônimo para Tabelas e Colunas. Nós estamos alegres que os dois puderam comparecer aqui. Esperamos que vocês possam esclarecer algumas confusões para a gente.

**Pseudônimo da Tabela:** Certamente, é ótimo estar aqui e você pode nos chamar de PT (Pseudônimo da Tabela) e PC (Pseudônimo da Coluna) como abreviação durante esta entrevista [risos].

**Use a cabeça:** Ha ha! Isto certamente seria apropriado. Ok, PC, vamos começar com você. Por que todo este segredo? Você está tentando esconder alguma coisa?

**Pseudônimo da Coluna:** De forma alguma! Se estiver fazendo algo, estou tentado é deixar as coisas mais compreensíveis. Creio que falo por nós dois, certo PT?

**TA:** Sim, está. No caso de PC, já deveria estar bem claro o que ele está tentando fazer. Ele pega nomes de colunas longos e redundantes e os faz mais fáceis de serem acompanhados, mais acessíveis. Ele também dá tabelas resultado com nomes de colunas bem úteis. Minha história é um pouco diferente.

**Use a Cabeça:** Tenho que admitir, você não é tão conhecido PT. Eu já vi o modo como você opera, mas ainda não tenho certeza do que faz. Você não se aparece nenhum pouco nos resultados quando te usamos em uma consulta.

**PT:** Sim, é verdade. Mas acho que você ainda não me viu fazer o meu maior chamado.

**Use a cabeça:** Maior chamado? Parece intrigante. Continue.

**PT:** Eu existo para fazer as conexões mais fáceis de serem escritas.

**PC:** E você me ajuda também nestas mesmas juntas, PT.

**Use a cabeça:** Eu não estou entendendo, você pode me dar um exemplo?

**PT:** Posso ainda mostrar a sintaxe. Eu acho que vai ser bem claro o que é que ando fazendo:

```
SELECT mc.sobrenome, mc.primeiro_nome, p.profissão  
FROM meus_contatos AS mc  
INNER JOIN  
profissão AS p  
WHERE mc.id_contato = p.id;
```

**Use a cabeça:** Eu te vejo! Em todos os lugares que eu iria digitar meus\_contatos posso digitar apenas mc. E p para profissão. Muito mais simples e bem útil quando tenho que incluir dois nomes de tabela em uma única consulta.

**PT:** Especialmente quando as tabelas possuem nomes similares. Fazer suas consultas mais fáceis de serem compreendidas, não só lhe ajuda a escrevê-las, mas também ajuda a lembrar o que elas estão fazendo quando você retorna para vê-las no futuro.

**Use a cabeça:** Muito obrigado, PT e PC. Isto foi...ahh... onde eles foram?



## Sua caixa de ferramentas SQL

Você acabou de completar o capítulo 8 e pode fazer conexões como um profissional em SQL. Cheque todas as técnicas que aprendeu. Para uma lista completa de dicas neste livro, veja o Apêndice iii.

### CONEXÃO INTERNA

Qualquer conexão que combina os registros de duas tabelas utilizando alguma condição.

### CONEXÃO NATURAL

Uma conexão interna que não utiliza a cláusula 'ON'. Ela só funciona se você estiver conectando duas tabelas com nomes idênticos.

### EQUIJOIN E NÃO-EQUIJOIN

Ambas são conexões internas. A EQUIJOIN retorna linhas que são iguais e a NÃO-EQUIJOIN retorna quaisquer linhas que não sejam iguais.

### CONEXÃO CRUZADA

Retorna cada linha de uma tabela cruzada com qualquer linha de uma segunda tabela. Conhecida por muitos como CONEXÃO CARTESIANA e SEM CONEXÃO.

### CONEXÃO VÍRGULA

A mesma coisa que a CONEXÃO CRUZADA, exceto que uma vírgula é utilizada ao invés das palavras-chave CROSS JOIN.



Aponte seu lápis

Solução

Página 281.

Você já sabe como utilizar o comando ALTER a esta altura, então você precisa de ALTER (alterar) meus\_contatos para ter quatro novas colunas. Nomeie-as interesse1, interesse2, interesse3, interesse4.

`ALTER TABLE meus_contatos`

`ADD (interesse1 VARCHAR(20), interesse2 VARCHAR(20),  
interesse3 VARCHAR(20), interesse4 VARCHAR(20));`



Aponte seu lápis

Solução

Preencha as lacunas para completar o comando UPDATE de Greg. Nós demos algumas observações para lhe ajudar.

A diferença entre `SUBSTRING_INDEX` e `SUBSTR` é que `SUBSTRING_INDEX` está procurando por uma string \*dentro\* da coluna interesses - neste caso, uma vírgula - e retornando tudo que está na frente dela. `SUBSTR` está encurtando o tamanho da coluna interesse - começando logo após o primeiro interesse, uma vírgula, espaço (que é 2 caracteres a mais +2) - ao final da string.

`UPDATE meus_contatos SET`

`interesse1 = SUBSTRING_INDEX(interesses, ',', 1),`  
`interesses = SUBSTR(interesses, LENGTH(interesse1)+2),`  
`interesse2 = SUBSTRING_INDEX(..., ',', 1),`  
`interesses = SUBSTR(..., LENGTH(interesse2)+2),`  
`interesse3 = SUBSTRING_INDEX(..., ',', 1),`  
`interesses = SUBSTR(..., LENGTH(interesse3)+2),`  
`interesse4 = ...`

Depois de ter removido os três primeiros interesses da coluna interesses, tudo que sobrou foi o quarto interesse. Esta linha basta ser movida para a nova coluna. A esta altura, poderíamos simplesmente renomeá-la como coluna `intereesse4`, ao invés de criar uma nova.

A coluna interesses está vazia depois de termos executado o comando.

Interesses	intereesse1	intereesse2	intereesse3	intereesse4
primeiro, segundo, terceiro, quarto	primeiro	segundo	terceiro	quarto

## 9 subconsultas

# Consultas dentro de consultas \*



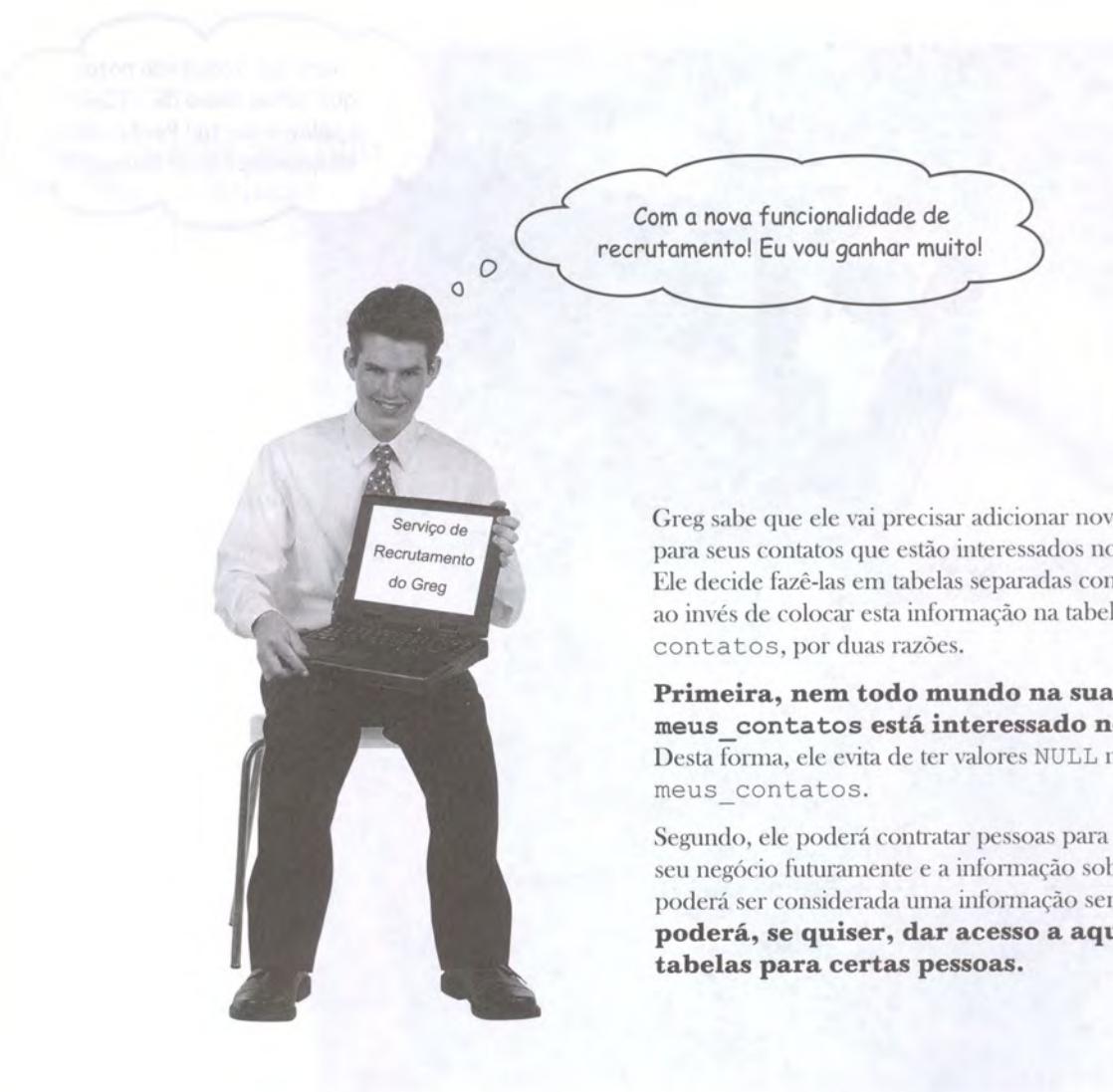
Será que todos vão notar  
que estou cheia de... (Qual  
a palavra certa? Perfeição?  
Resplandecência? Beleza?)

**Sim, Jack, eu gostaria de uma pergunta de duas partes, por favor.** Conexões são ótimas, mas às vezes você precisa perguntar ao seu banco de dados mais que uma pergunta. Ou pegar o resultado de uma consulta e usá-lo como entrada para outra consulta. É aí que as subconsultas entram. Elas irão ajudar a evitar dados duplicados, fazer suas consultas mais dinâmicas, e ainda fazer com que você tenha acesso aos concertos caríssimos depois da festa. (Bem, na verdade não, mas duas de três coisas já não é nada mal!)

## Greg entra no negócio de recrutamento laboral

Até então, o banco de dados `gregs_list` tem sido um trabalho de amor. Ele ajudou Greg a encontrar pares para seus amigos, mas ele ainda não ganhou nenhum dinheiro com isso.

Ele então pensou que poderia iniciar um negócio de recrutamento onde ele combina seus contatos com possíveis empregos.



Greg sabe que ele vai precisar adicionar novas tabelas para seus contatos que estão interessados no serviço. Ele decide fazê-las em tabelas separadas com um-a-um ao invés de colocar esta informação na tabela `meus_contatos`, por duas razões.

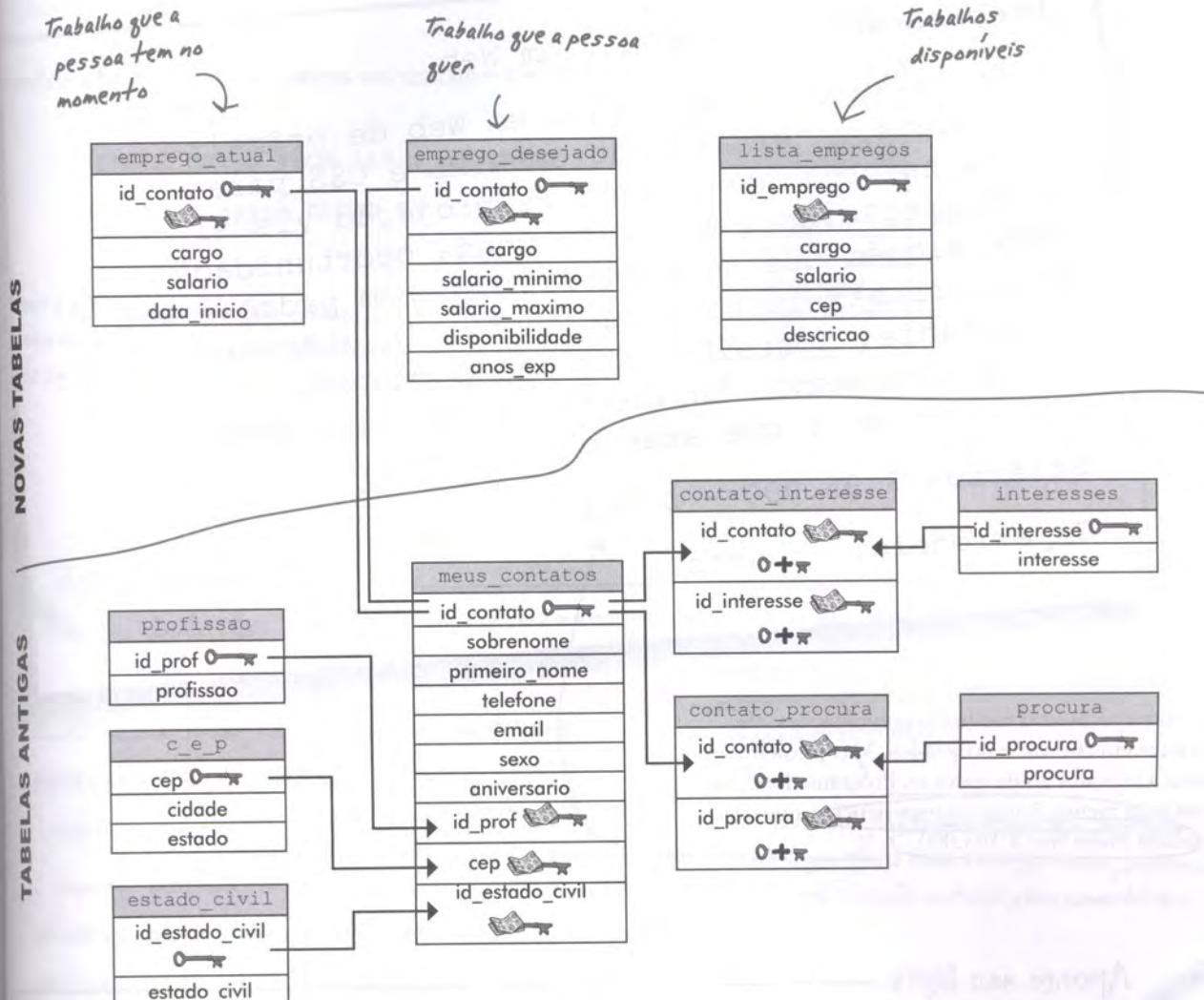
**Primeira, nem todo mundo na sua lista de `meus_contatos` está interessado no serviço.** Desta forma, ele evita de ter valores NULL na tabela `meus_contatos`.

Segundo, ele poderá contratar pessoas para ajudá-lo em seu negócio futuramente e a informação sobre salário poderá ser considerada uma informação sensível. **Ele poderá, se quiser, dar acesso a aquelas tabelas para certas pessoas.**

## gregs\_list fica com mais tabelas

adicionou novas tabelas a seu banco de dados para rastrear informações acerca da posição desejada e faixa expectativa salarial, bem como posição atual e salário. Ele também criou uma tabela simples para armazenar a informação das listas de empregos.

TABELAS ANTIGAS



Uma vez que as duas novas tabelas possuem um relacionamento um-a-um com `meus_contatos`, ele foi capaz de utilizar conexões naturais até agora com grande sucesso e facilidade.

## Greg utiliza uma conexão interior

Greg conseguiu uma lista de empregos excelente, e ele está tentando encaixar pessoas em seu banco de dados. Ele quer encontrar a melhor combinação para o emprego já que vai receber uma taxa se o seu candidato for contratado.

**Precisa-se:** Programador em Web

Procura-se um Programador em Web de primeira qualidade e certificado em HTML e CSS para trabalhar com nosso time de projeção visual e interação. Esta é uma tremenda oportunidade para alguém que seja meticoloso com padrões web para brilhar junto a uma companhia visivelmente importante. Trabalhar com uma companhia maravilhosamente influente operada por pessoas inteligentes e que amam o que fazem.

**Salário:** \$ 95.000 - \$ 105.000

**Experiência:** +5 anos

Após encontrar as poucas melhores combinações, ele poderá entrar em contato e projetá-los. Mas primeiro, ele quer puxar a informação de todos os Programadores em Web com pelo menos 5 anos de experiência e que não requer salário maior que \$ 105.000.



Aponte seu lápis

Escreva a consulta para encontrar os candidatos qualificados do banco de dados.

emprego_atual
id_contato
cargo
salario
data_inicio

Este é o menor salário que eles vão aceitar por um novo emprego.

Este é salário pelo qual eles almejam em um novo emprego.

emprego_desejado
id_contato
cargo
salario_minimo
salario_maximo
disponibilidade
anos_exp

lista_empregos
id_emprego
cargo
salario
cep
descricao

## mas ele quer tentar algumas outras consultas

Ele tem mais vagas de emprego do que ele pode preencher. Ele quer procurar por pessoas na sua tabela profissão para ver se ele pode encontrar qualquer combinação para as vagas de sua lista de empregos. Então, poderá fazer uma conexão natural com meus contatos para obter a informação de seus contatos e verificar se eles possuem interesse.

Primeiro ele seleciona todos os cargos de sua tabela emprego\_atual.

```
SELECT cargo FROM lista_empregos
GROUP BY cargo ORDER BY cargo;
```

↑  
Usamos o GROUP By para que tenhamos  
uma linha para cada cargo. Nós também usamos em ordem alfabética.

Os resultados

cargo
Cozinheira
Cabeleireira
Garçom
Web designer
Programador em Web

← Estes são apenas  
alguns dos cargos  
na tabela do Greg  
emprego\_atual.

Aponte seu lápis

Solução

Escreva a consulta para encontrar os candidatos qualificados do banco de dados.

```
SELECT mc.sobrenome, mc.primeiro_nome, mc.telefone
FROM meus_contatos AS mc
NATURAL JOIN
    emprego_desejado AS ed
WHERE ed.cargo = 'Programador em Web'
AND ed.salario_minimo < 105000;
```

← Precisamos apenas da informação do contato, já que  
nós sabemos que eles procuram por empregos como  
Programadores em Web.

← Já que ambas as tabelas meus\_contatos e emprego\_desejado  
compartilham a coluna id\_contato como chave primária, podemos  
simplesmente utilizar uma conexão natural para conectá-las.

↑ Estamos interessados apenas em pessoas que levarão em conta o salário. Nós  
olhamos para o salario\_minimo para descobrir se o salário oferecido é maior que o  
mínimo que a pessoa aceitará.

Agora Greg utiliza a palavra-chave IN para verificar se tem alguma combinação para esta vaga de emprego ao longo de seus contatos

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo  
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc  
WHERE  
ea.cargo IN ('Cozinheira', 'Cabeleireira', 'Garçom', 'Web designer', 'Programador em Web');
```

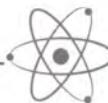
Ainda se lembra da palavra-chave IN? Ela retorna uma linha se ea.cargo está no grupo de cargos em parênteses.

Resultados da primeira pesquisa.

Funciona!

mc.primeiro_nome	mc.sobrenome	mc.telefone	ea.cargo
Joe	Lonnigan	(555) 555-3214	Cozinheira
Wendy	Hillerman	(555) 555-8976	Garçom
Sean	Miller	(555) 555-4443	Web designer
Jared	Callaway	(555) 555-5674	Programador em Web
Juan	Garza	(555) 555-0098	Programador em Web

Mas ele ainda está tendo que digitar em duas consultas separadas



PODER DO  
CÉREBRO

Tente combinar as duas consultas em uma só consulta. Escreva a consulta única aqui.

## Subconsultas

Para conseguir fazer o que aquelas duas consultas fazem, em uma só consulta, precisamos adicionar uma subconsulta na nossa consulta.

Vamos chamá-la de *segunda* consulta que usamos para obter as combinações entre a tabela profissão e a consulta EXTERNA, porque ela cobrirá dentro de si a consulta INTERNA. Vamos ver como isso funciona.

### Consulta Externa

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo  
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc  
WHERE  
ea.cargo IN ('Cozinheira', 'Cabeleireira', 'Garçom', 'Web designer', 'Programador em Web');
```

Esta parte é a consulta externa.

Esta parte pode ser removida e  
recolocada como parte de nossa  
primeira consulta, o que se tornará  
a consulta interna.

Todas essas profissões acima, em parênteses, vêm da primeira consulta que fizemos, aquela que selecionou todos os cargos da tabela emprego\_atual. Então - essa é a parte inteligente, então preste atenção - nós podemos trocar de lugar aquela parte da consulta interna com parte da primeira consulta. Isto ainda vai produzir todos os resultados entre os parênteses acima, mas esta consulta agora fica integrada como a subconsulta:

Uma subconsulta é  
uma consulta envolvida  
por outra consulta.  
Também é chamada de  
consulta INTERNA.

### Consulta interna

```
SELECT cargo FROM lista_empregos  
GROUP BY cargo ORDER BY cargo;
```

← Esta parte da primeira consulta se tornará  
a consulta interna ou subconsulta.

## Nós combinamos as duas em uma consulta com uma subconsulta

Tudo o que fizemos foi combinar as duas consultas em uma só. A primeira consulta é conhecida como consulta externa. A outra dentro dela é conhecida como consulta interna.

Consulta Externa

+

Consulta interna

=

Consulta com subconsulta

*Consulta externa query*

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
WHERE ea.cargo IN (SELECT cargo FROM lista_empregos);
```

*As duas consultas combinadas em uma só é uma consulta contendo subconsulta.*

*Não precisamos digitar novamente todas as profissões da nossa primeira consulta porque a consulta interna já tomou conta disto por nós!*

E estes são os resultados obtidos quando executamos nossa tabela, precisamente os mesmos resultados de quando digitamos todas as profissões na cláusula WHERE, mas com menos digitação.

*Mesmos resultados que anteriormente só que utilizando uma só consulta!*

mc.primeiro_nome	mc.sobrenome	mc.telefone	ea.cargo
Joe	Lonnigan	(555) 555-3214	Cozinheira
Wendy	Hillerman	(555) 555-8976	Garçom
Sean	Miller	(555) 555-4443	Web designer
Jared	Callaway	(555) 555-5674	Programador em Web
Juan	Garza	(555) 555-0098	Programador em Web



Anatomia de uma consulta dentro de uma consulta

## Como se uma consulta já não fosse o bastante, conheça a subconsulta

### Uma subconsulta não é nada mais que uma consulta dentro de outra consulta

A consulta externa é conhecida como consulta contentora ou de consulta externa. A consulta de dentro é a consulta interna ou subconsulta.

```
SELECT alguma_coluna, outra_coluna
FROM tabela
WHERE coluna = (SELECT coluna FROM tabela);
```

*Consulta externa. Às vezes conhecida como consulta contentora.*

```
SELECT alguma_coluna, outra_coluna
FROM tabela
WHERE coluna = (SELECT coluna FROM tabela);
```

*Consulta externa.*

*Consulta interna ou subconsulta.*

*Consulta interna.*

VALOR

*Nossa subconsulta retorna um valor escalar (uma coluna, uma linha), que é então comparada com as colunas na cláusula WHERE.*

## Uma subconsulta em ação

Vamos ver uma consulta comparativa em ação a partir da tabela `meus_contatos`. Primeiro seu Sistema SQL da tabela `c_e_p`, daí da compara aquele valor às colunas na cláusula WHERE.

```
(SELECT cep FROM
c_e_p WHERE cidade =
'Memphis' AND estado = 'TN')
```



Anatomia de uma  
consulta dentro de  
uma consulta  
(Continuação)

VALOR

```
SELECT sobrenome, primeiro_nome  
FROM meus_contatos  
WHERE cep =  
      (SELECT cep FROM  
       c_e_p WHERE cidade =  
       'Memphis' AND estado = 'TN')
```

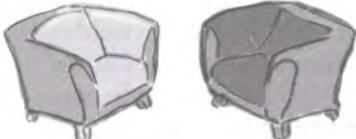
Esta consulta seleciona  
o nome das pessoas de  
Memphis, Tennessee.

não existem  
Perguntas Idiotas

Você pode fazer a mesma consulta que fez acima, desta forma:

```
SELECT sobrenome, primeiro_nome  
FROM meus_contatos mc  
NATURAL JOIN c_e_p c  
WHERE c.cidade = 'Memphis'  
AND c.estado = 'TN'
```

## Bate-papo



Conversa de hoje: Você é uma externa ou uma interna?

### Consulta externa

Sabe, eu realmente não preciso de você, Consulta Interna. Eu estaria muito bem sem você.

U-hu! Você me deu apenas um resultado. Usuários querem dados, e muitos. Eu dou isso a eles. Por que aposte que se você não estivesse lá, eles seriam ainda mais gratos.

Não se eu adicionar uma cláusula WHERE

### Consulta interna

Eu poderia ficar na minha também. Você acha que é divertido dar a você um resultado específico definido apenas para entregá-lo e transformá-lo em uma porção de linhas que se combinam. Quantidade não é qualidade, você sabe.

Não, eu dei aos seus resultados um pouco de propósito, sem mim. Sem mim, você poderia espalhar todos os dados na tabela.

**Consulta externa**

Ah...sim, precisa sim. Que bem faria uma resposta de uma só linha e uma só coluna? Elas não são informações suficientes.

Claro, mas eu fico sozinho.

**Consulta interna**

Então é isso, EU SOU sua cláusula WHERE, e uma bem específica, se assim posso dizer, além de mim mesmo. De fato, eu não preciso para nada.

Então talvez possamos trabalhar bem juntos. Eu dou a direção dos seus resultados

Assim como eu, a maioria das vezes.

**Regras para subconsultas**

Existem algumas regras para todas as subconsultas seguintes. Preencha os espaços em branco com as palavras abaixo (você pode precisar de algumas por mais de uma vez).

PONTO-E-VÍRGULA

SELECT

FROM

UPDATE

PARENTESES

COLUMN LIST

HAVING

INSERT

END

DELETE

*Livro da Ordem do SQL*

Uma subconsulta sempre é um comando .....

Subconsultas estão sempre dentro dos .....

Subconsultas não têm seus próprios .....  
Como sempre, um ..... vai ao ..... da consulta inteira.

*Livro da Ordem do SQL*

Subconsultas podem aparecer quatro vezes em uma consulta:

Cláusula ..... , SELECT ..... como uma das colunas, cláusula ..... , e em uma ..... cláusula .....

Subconsultas podem ser

utilizadas com ..... e é ..... claro, .....



## Regras para subconsultas

Existem algumas regras para todas as subconsultas seguintes. Preencha os espaços em branco com as palavras abaixo (você pode precisar de algumas por mais de uma vez).

### Livro da Ordem do SQL

Uma subconsulta sempre é um comando SELECT.....

Subconsultas estão sempre dentro dos PARENTESES.....

Subconsultas não têm seus próprios PONTO-E-VÍRGULAS. Como sempre, um PONTO-E-VÍRGULAS vai ao FINAL da consulta inteira.

### Livro da Ordem do SQL

Subconsultas podem aparecer quatro vezes em uma consulta: Cláusula SELECT....., SELECT COLUMNLIST..... como uma das colunas, cláusula FROM....., e em uma cláusula HAVING.....

Subconsultas podem ser utilizadas com INSERT....., DELETE....., UPDATE..... e é claro, SELECT.....

---

### não existem Perguntas Idiotas

---

**P:** O que é permitido a uma consulta interna retornar? E a consulta externa?

**R:** Na maioria dos casos, a consulta interna pode retornar apenas um só valor – isto é, uma coluna com uma linha. A consulta externa pode pegar aquele valor e usá-lo para comparar contra todos os valores em uma coluna.

**P:** Por que você diz “um só valor” quando o exemplo na página 314 retorna a coluna inteira cheia de valores?

**R:** Porque o operador IN está procurando em um conjunto de valores. Se você utilizar um operador de comparação, como o sinal de = na anatomia, poderá ter apenas um valor para ser comparado com todos

os valores em sua coluna.

**P:** Ainda não estou certo se uma subconsulta pode retornar um só valor ou mais que um valor. Quais são as regras oficiais?

**R:** Em geral, uma subconsulta deve retornar um valor único. IN é a exceção. Na maioria das vezes, subconsultas precisam retornar um só valor para funcionar.

**P:** Então o que acontece se sua subconsulta retornar mais do que um valor, mas não estiver utilizando uma cláusula WHERE que contenha um conjunto de valores?

**R:** Caos! Destrução em massa! Na verdade, você vai somente obter um erro.



Sim, estas regras são legais, ou sei lá, mas o que quero saber é como posso dar um fim naqueles nomes longos na minha coluna resultados, como mc.sobrenome. Existe uma regra para isso?

### **Na verdade, há duas coisas que você pode fazer para ajudar a diminuir a interferência.**

Você pode criar pseudônimos para suas colunas na sua lista SELECT. A tabela que receberá com seus resultados é, de repente, mais clara.

Aqui está a subconsulta que criamos, mas com pseudônimos bem curtos para colunas

Daremos a coluna `primeiro_nome` da `meus_contatos` um pseudônimo de `primeironome` em nossos resultados.

...e a coluna `sobrenome` da `meus_contatos` terá um pseudônimo `sobrenome` em nossos resultados.

```
SELECT mc.primeiro_nome AS primeironome, mc.sobrenome AS sobrenome,
mc.telefone AS fone, ea.cargo AS cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
WHERE cargo IN (SELECT cargo FROM lista_empregos);
```

Aqui estão os resultados que a consulta nos deu.

Perceba como ao utilizar o pseudônimo da coluna faz os resultados muito mais fáceis de serem entendidos.

E já que pseudônimos são temporários, não estamos afetando nenhum nome de nossas tabelas ou colunas em nenhuma das tabelas.

Lembre-se, a palavra-chave `AS` é opcional, então você pode deixá-la de fora ao criar seus pseudônimos.

mc.primeiro_nome	mc.sobrenome	mc.telefone	ea.cargo
Joe	Lonnigan	(555) 555-3214	Cozinheira
Wendy	Hillerman	(555) 555-8976	Garçom
Sean	Miller	(555) 555-4443	Web designer
Jared	Callaway	(555) 555-5674	Programador em Web
Juan	Garza	(555) 555-0098	Programador em Web

## **Um acompanhamento na construção de uma subconsulta**

Uma complicada sobre subconsultas não é a estrutura; é descobrir qual parte da consulta é de uma subconsulta. Ou mesmo se você realmente precisa de uma.

Analizar consultas é muito parecido com descobrir problemas nas palavras. Você identifica as palavras na questão que combinam com coisas que você sabe do que se trata (como os nomes das tabelas e colunas) e que as separa.

Vamos passar pela análise de uma questão que queremos fazer para o banco de dados e como fazer uma consulta de tudo isto. Primeiro, a questão:



Quem ganha mais de toda a tabela meus\_contatos?

### Disse que a pergunta

Reformule a pergunta em termos das tabelas e colunas de seu banco de dados.

"Quem" quer dizer que você quer o primeiro nome e o sobrenome da tabela meus\_contatos.

"Ganha mais" quer dizer que você precisa do valor MAX da sua tabela emprego\_atual.

### Quem ganha mais dinheiro de toda a tabela meus\_contatos?

primeiro\_nome e sobrenome da tabela meus\_contatos

MAX (salario) da tabela emprego\_atual.

### Identifique uma consulta que responda parte da questão

Já que estamos criando uma subconsulta não-correlacionada, podemos dividir nossa pergunta e construir uma consulta que responda apenas parte da pergunta.

Aquele MAX (salario) parece um bom candidato para nossa primeira consulta.

`SELECT MAX(salario) FROM emprego_atual;`

Ainda se lembra do MAX? Ele retorna o maior valor da coluna indicada entre parênteses.

### Continue dissecando sua consulta

A primeira parte da consulta também é fácil; nós só precisamos selecionar o primeiro nome e o sobrenome:

`SELECT mc.primeiro_nome, mc.sobrenome  
FROM meus_contatos AS mc;`

Selecionar o primeiro nome e o sobrenome

### Finalmente, descubra como relacionar as duas

Nós não só precisamos dos nomes das pessoas da meus\_contatos, nós precisamos saber seus salários para que possamos compará-los para nosso MAX (salario). Precisamos de uma conexão natural para puxar a informação do salário de cada pessoa:

`SELECT mc.primeiro_nome,  
mc.sobrenome, ea.salario  
FROM meus_contatos AS mc  
NATURAL JOIN emprego_atual AS ea;`

Use uma conexão natural para obter o salário de cada um

**E agora adicione a cláusula WHERE para conectá-las**

Criamos uma grande consulta que responde a questão, "quem ganha mais dinheiro?".

*Aqui está a parte que acabamos de fazer - ela obtém o salário de cada pessoa.*

```
SELECT mc.primeiro_nome, mc.sobrenome, ea.salario
FROM meus_contatos AS mc NATURAL JOIN emprego_atual AS ea
WHERE ea.salario =
(SELECT MAX (ea.salario) FROM emprego_atual);
```

*É aqui que está a primeira parte que agora é nossa subconsulta para encontrar o valor MAX do salário. O valor desta subconsulta é comparado com a parte externa da consulta para obter os resultados.*

mc.primeiro_nome	mc.sobrenome	ea.salario
Mike	Scala	187000

*Então é o Mike? Eu deveria saber. Ele nunca pega o cheque.*



*Parece que poderíamos fazer tudo isso sem uma subconsulta.*

**É verdade, a subconsulta não é o único jeito de fazer isso**

Você poderia ter feito a mesma coisa utilizando uma conexão interna e o comando LIMIT. Como muitas outras coisas em SQL, há mais de um jeito de fazer isso.

**PODER DO CÉREBRO**

Escreva outra consulta para descobrir quem ganha mais dinheiro dentre todos os contatos de Greg.

*Não me importo se existem múltiplas maneiras de fazer a mesma coisa. Eu quero saber o melhor jeito, ou pelo menos uma razão para escolher entre um e outro.*

**Bem observado**

Por que você não olha a entrevista SQL revelado na página 325?

## Uma subconsulta como uma coluna no comando SELECT

Uma subconsulta pode ser usada como uma das colunas no comando SELECT. Observe esta consulta:

```
SELECT mc.primeiro_nome, mc.sobrenome,
       (SELECT estado
        FROM c_e_p
        WHERE mc.cep = cep) AS estado
     FROM meus_contatos mc;
```

*Estamos definindo o pseudônimo para uma coluna, estado.*

Nós podemos dissecar esta consulta ao olhar para a subconsulta. A subconsulta simplesmente combina o cep ao estado correspondente na tabela `c_e_p`.

Em termos simples, a consulta acima está fazendo isto:

Passe por todas as linhas da `meus_contatos`. Para cada uma, puxe o primeiro nome, o sobrenome e o estado (onde colocamos o cep e combinando com o estado correspondente na tabela `c_e_p`).

Lembre-se que a subconsulta pode retornar apenas um valor único, então cada vez que ela é executada, uma linha é retornada. Aqui estão como alguns dos resultados desta consulta podem ser:

mc.primeiro_nome	mc.sobrenome	estado
Joe	Lonnigan	TX
Wendy	Hillerman	CA
Sean	Miller	NY
Jared	Callaway	NJ
Juan	Garza	CA

**Se uma subconsulta é utilizada como a expressão de uma coluna em um comando SELECT, ela poderá retornar apenas um valor de uma coluna.**

Andy,  
disse a  
sua tal  
email

## Outro exemplo: Subconsulta com uma conexão natural

Andy, amigo de Greg, vem reclamando do ótimo salário que recebe. Ele não se lembra quanto era, mas Greg acredita que ele tenha esta informação em sua tabela. Ele faz uma consulta natural rápida para encontrá-la, utilizando o email de Andy.

```
SELECT ea.salario
FROM meus_contatos mc NATURAL JOIN emprego_atual ea
WHERE email = 'andy@weatherorama.com';
```

*← Esta consulta retornará o salário de Andy, ou seja, um só valor.*

*← Esta será a consulta interna.*

Greg percebe que sua consulta retornará um único valor. Ao invés de executar o comando, pegar o valor e jogar toda a informação novamente em outra consulta, ele decide transformá-la em uma subconsulta.

Ele escreve uma consulta simples que:

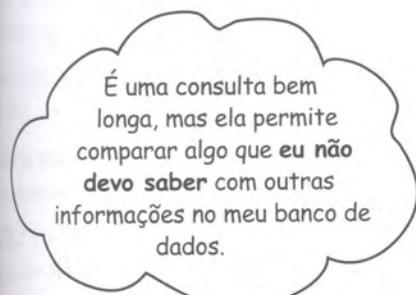
- obtém o salário de Andy e
- compara-o com outros salários
- e retorna o primeiro nome e sobrenome das pessoas com seus salários
- daqueles que ganham mais que Andy

*← Este comando usará o operador de comparação*

*← Salários maiores que o de Andy.*

Aqui está a consulta externa:

```
SELECT mc.primeiro_nome, mc.sobrenome, ea.salario
FROM
meus_contatos AS mc NATURAL JOIN emprego_atual AS ea
WHERE
ea.salario > (O SALÁRIO DE ANDY VAI AQUI)
```



## Uma consulta não-correlacionada

Quando juntamos os pedaços, aqui está a consulta completa. Primeiro o Sistema processa a consulta interna, depois ele usa o resultado para descobrir o resultado da consulta externa.

O Sistema SQL processa essa parte primeiro.

Apenas mostre as pessoas que têm o salário maior que o de Andy.

Estas duas consultas são processadas separadamente pelo Sistema SQL.

Aqui estão um pouco dos resultados. Nós não utilizamos um ORDER BY, então eles não estão em nenhuma ordem.

mc.primeiro_nome	ms.sobrenome	ea.salario
Gus	Logan	46500
Bruce	Hill	78000
Teresa	Semel	48000
Randy	Wright	49000
Julie	Moore	120000

Todas as subconsultas que você viu até agora são conhecidas como subconsultas não-correlacionadas. A consulta interna é processada primeiro, então o resultado é usado na cláusula WHERE da consulta externa. **Entretanto a consulta interna não depende de nenhuma forma da consulta externa; ela pode ser executada como uma consulta independente.**

Consulta externa

Consulta interna

Consulta externa é processada em segundo. Seu resultado depende do valor vindo da consulta externa.

Consulta interna funciona sozinha e é processada primeiro.

Obtenha o primeiro nome, sobrenome e o salário.

```
SELECT mc.primeiro_nome, mc.sobrenome, ea.salario
FROM
meus_contatos AS mc NATURAL JOIN emprego_atual AS ea
WHERE
ea.salario > (SELECT ea.salario
FROM meus_contatos mc NATURAL JOIN emprego_atual ea
WHERE email = 'andy@weatherorama.com');
```

A subconsulta que obtém o salário de Andy para a consulta externa para usar como comparação.

Esta é processada em segundo lugar.

Se a consulta funciona sozinha e não se refere a nada da consulta externa, ela é uma subconsulta não-correlacionada.

(se você pode manejá-lo, é impressionante)



## SQL Revelado

A entrevista desta semana:  
Escolhendo a melhor maneira para consultar  
quando você tem mais que uma opção

**Use a Cabeça:** Bem vinda SQL, nós agradecemos pela entrevista pessoal. Nós sabemos que as coisas tem andado difíceis.

**SQL:** Difícil? É assim que você chama isso? Eu digo que as coisas têm estado atribuladas, perturbadoras e muito difícil de quantificar e, ao mesmo tempo, ser muito enrolada.

**Use a cabeça:** Ah, certo! Esse é o ponto aqui. Você está recebendo reclamações de que talvez não seja muito flexível. Você nos dá muitas opções quando fazemos algumas perguntas.

**SQL:** Admito que sou flexível, que você pode fazer a mesma pergunta uma porção de vezes e eu darei a mesma resposta.

**Use a cabeça:** Algumas pessoas diriam que você não tem opinião própria.

**SQL:** Me recuso a ficar na defensiva por causa disso. Não sou vilão aqui.

**Use a cabeça:** Não, sabemos que você não é, é que você é... impreciso.

**SQL:** Ha! Eu, impreciso! Já tive o suficiente sobre isso (levantando).

**Use a cabeça:** Não, não vá. Nós só queremos algumas respostas. Às vezes você nos permite perguntar a mesma coisa em várias formas diferentes.

**SQL:** E o que há de errado nisso?

**Use a cabeça:** Nada, na verdade. Só queremos saber O QUE deveríamos perguntar. Isso importa se você nos der as mesmas respostas?

**SQL:** Claro que importa! Às vezes você me pergunta alguma coisa, e isso leva um tempão para responder. Às vezes, BANG, e estou pronto. O verdadeiro objetivo é me perguntar da maneira certa.

**Use a cabeça:** Então tem a ver com quanto tempo demora em responder? É assim que escolhemos como te perguntar?

**SQL:** Bem, amigo. Mas é claro que sim. Tem tudo a ver com que você me perguntou. Estou aqui para tentar responder as suas perguntas, quando elas estão corretas.

**Use a cabeça:** Velocidade? Este é o segredo?

**SQL:** Olhe, vou dar uma pista. O segredo sobre os bancos de dados é a palavra-chave GROW. Você quer que suas perguntas sejam as mais fáceis possíveis de serem respondidas. Porque se você perguntar "Whodunnit", eu vou precisar que me faça pensar sobre isto o mínimo possível. Mande questões fáceis que eu darei respostas rápidas.

**Use a cabeça:** Entendi, mas como sabemos quais são as questões fáceis?

**SQL:** Bom, para iniciantes, consultas cruzadas são uma perda de tempo e as subconsultas correlacionais estão do mesmo lado, no time dos lentos.

**Use a cabeça:** Algo mais?

**SQL:** Bem...

**Use a cabeça:** Por favor, continue.

**SQL:** Experimente. Às vezes seu melhor palpite é criar tabelas de testes e tentar diversas consultas, daí você poderá comparar quanto tempo cada uma gasta. Ah, e conexões são mais eficientes que subconsultas.

**Use a cabeça:** Obrigado, SQL. Nem acredito que esse é o grande segredo...

**SQL:** Sim, obrigado por gastar meu tempo.

# WORKSHOP CONSTRUA UMA SUBCONSULTA

Leia as hipóteses abaixo. Siga as instruções para escrever as duas consultas como foi pedido, depois as combine em uma subconsulta.

- 1.** Greg quer verificar qual a média do salário para Programadores em Web na sua tabela `emprego_atual`. Depois ele quer verificar o quanto as pessoas estão ganhando em relação ao salário médio para aquele cargo. Se ele encontrar pessoas ganhando abaixo da média, ele pode usá-las como foco porque elas podem estar interessadas em obter um novo emprego.

**Escreva uma consulta para obter o salário médio para o cargo de Programador em Web da tabela `emprego_atual`.**

.....  
.....  
.....

- 2.** Greg precisa do primeiro nome, sobrenome e salário de todos os Programadores em Web de sua tabela `emprego_atual`.

**Escreva uma consulta para obter o primeiro nome, sobrenome e salário de todos os Programadores em Web de sua tabela `emprego_atual`.**

.....  
.....  
.....

- 3.** Greg utilizou o salário médio (e um pouco de matemática) como uma subconsulta para exibir cada Programador em Web e quanto o salário deles está acima ou abaixo da média.

**Combine as duas consultas. Utilize a subconsulta como parte do comando `SELECT` para uma lista de colunas.**

.....  
.....  
.....

# SOLUÇÃO DO WORKSHOP

## CONSTRUA UMA SUBCONSULTA

Leia as hipóteses abaixo. Siga as instruções para escrever as duas consultas como foi pedido, depois as combine em uma subconsulta.

- 1** Greg quer verificar qual a média do salário para Programadores em Web na sua tabela `emprego_atual`. Depois ele quer verificar o quanto as pessoas estão ganhando em relação ao salário médio para aquele cargo. Se ele encontrar pessoas ganhando abaixo da média, ele pode usá-las como foco porque elas podem estar interessadas em obter um novo emprego.

Escreva uma consulta para obter o salário médio para o cargo de Programador em Web da tabela `emprego_atual`.

`SELECT AVG(salario) FROM emprego_atual WHERE cargo = 'Programador em Web';`

A palavra-chave `AVG` é exatamente a...  
que precisávamos aqui...

- 2** Greg precisa do primeiro nome, sobrenome e salário de todos os Programadores em Web de sua tabela `emprego_atual`.

Escreva uma consulta para obter o primeiro nome, sobrenome e salário de todos os Programadores em Web de sua tabela `emprego_atual`.

`SELECT mc.primeiro_nome, mc.sobrenome, ea.salario  
FROM meus_contatos mc NATURAL JOIN emprego_atual ea  
WHERE ea.cargo = 'Programador em Web';`

- 3** Greg utilizou o salário médio (e um pouco de matemática) como uma subconsulta para exibir cada Programador em Web e quanto o salário deles está acima ou abaixo da média.

Combine as duas consultas. Utilize a subconsulta como parte do comando `SELECT` para uma lista de colunas.

Aqui está nossa subconsulta.

`SELECT mc.primeiro_nome, mc.sobrenome, ea.salario,  
ea.salario - (SELECT AVG(salario) FROM emprego_atual WHERE cargo = 'Programador em Web')  
FROM meus_contatos mc NATURAL JOIN emprego_atual ea  
WHERE ea.cargo = 'Programador em Web';`

## Uma subconsulta não-correlacionada com múltiplos valores: IN, NOT IN

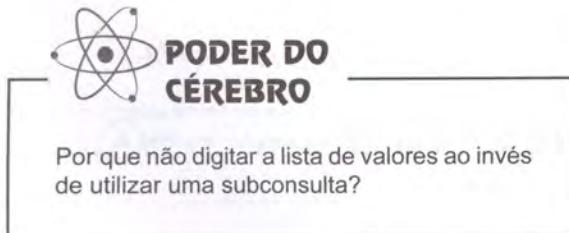
Leve em consideração a primeira consulta Greg tentou lá na página 313. Ela o ajudou a localizar as pessoas com o cargo que **combinava** com sua lista. Ela pega o conjunto completo de cargos retornados pelo SELECT na subconsulta e os compara com cada linha na tabela cargo\_atual para encontrar qualquer possível compatibilidade.

```
SELECT mc.primeiro_nome, mc.sobrenome, ea.cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
WHERE ea.cargo IN (SELECT cargo FROM lista_empregos);
    ↗
    IN avalia cada linha do ea.cargo em comparação com o conjunto inteiro retornado pela subconsulta.
```

Usar **NOT IN** ajudaria Greg visualizar os cargos que **não combinam** com sua lista. Ela pega o conjunto completo de cargos retornados pelo SELECT na subconsulta e avalia em comparação a cada linha da tabela emprego\_atual retornando qualquer valor *que não se ajuste* àqueles na tabela emprego\_atual. Agora Greg pode se dedicar em encontrar mais listas de empregos para aquele tipo de emprego.

```
SELECT mc.primeiro_nome, mc.sobrenome, ea.cargo
FROM emprego_atual ea NATURAL JOIN meus_contatos mc
WHERE ea.cargo NOT IN (SELECT cargo FROM lista_empregos);
    ↗
    NOT IN retorna qualquer cargo de emprego atual que não foram encontrados na listagem de empregos.
```

Estes tipos de consultas são chamadas de **subconsultas não-correlacionadas**, onde **IN** e **NOT IN** testam os resultados da subconsulta em comparação a consulta externa para ver se há resultados compatíveis ou não.



Uma subconsulta não-correlacionada usa IN ou NOT IN para testar se os valores retornados na subconsulta são integrantes de um conjunto (ou não).



## Exercícios

Escreva consultas com conexões e subconsultas não-correlacionadas, quando necessário, para responder as questões abaixo. Utilize o esquema do banco de dados gregs\_list para lhe ajudar.

*Diversos e iguais a este precisam agregar funções que você aprendeu com o Problema das Vendas de Biscoitos das Garotas Bandeirantes.*

- Liste os cargos que ganham salários iguais ao maior salário na tabela lista\_empregos.

→ Resposta na página 330.

- Liste o primeiro nome e o sobrenome das pessoas com o salário maior que o salário médio.

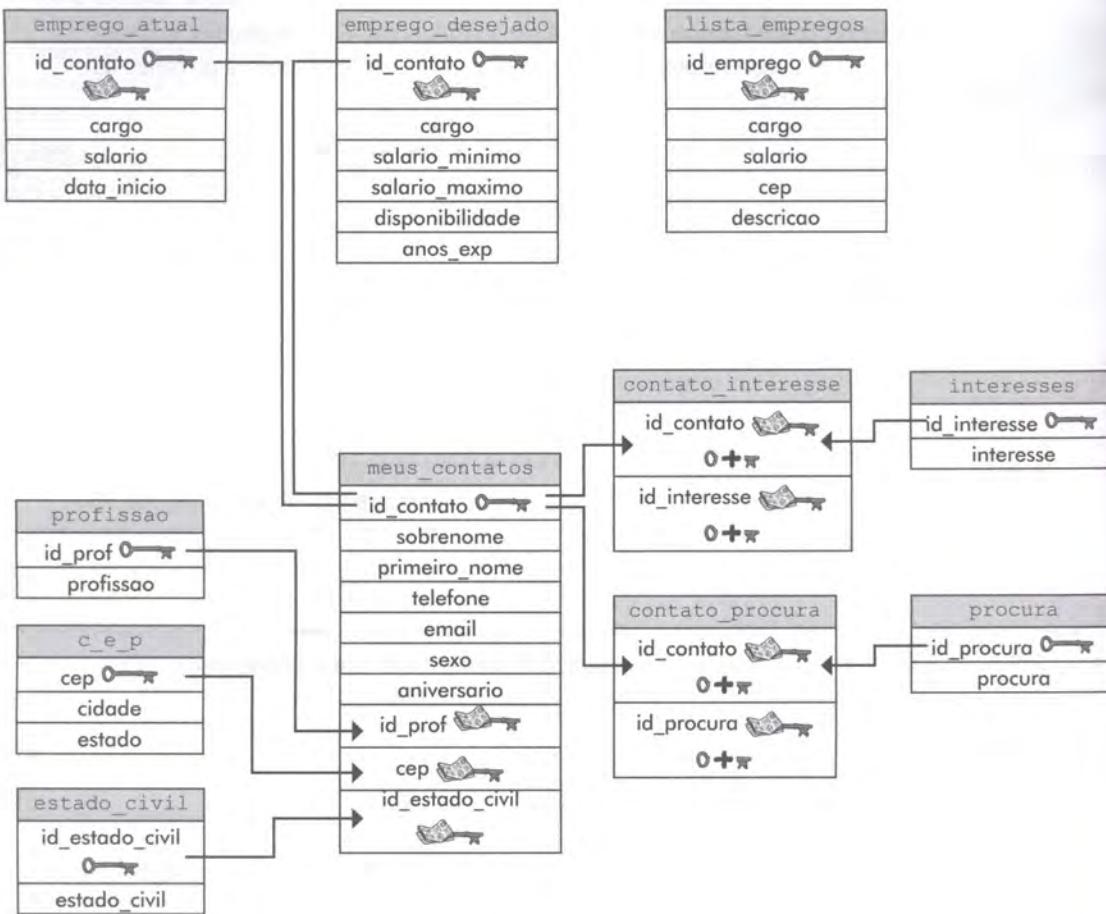
→ Respostas na página 331.

- Encontre todos os Web Designers que possuem o mesmo CEP que aqueles da tabela lista\_empregos para Web Designers.

→ Respostas na página 331.

- Liste qualquer pessoa que viva no mesmo CEP que a pessoa que atualmente possui o maior salário.

→ Respostas na página 331.



## Solução dos Exercícios

Escreva consultas com conexões e subconsultas não-correlacionadas, quando necessário, para responder as questões abaixo. Utilize o esquema do banco de dados `gregs_list` para lhe ajudar.

- Liste os cargos que ganham salários iguais ao maior salário na tabela lista\_empregos.

A consulta externa combina com o valor MAX do salário.

```
SELECT cargo FROM lista_empregos  
WHERE salario = (SELECT MAX(salario)  
                  FROM lista_empregos);
```

A subconsulta  
retorna um valor.

MAX retorna o maior valor de salário na tabela.

Liste o primeiro nome e o sobrenome das pessoas com o salário maior que o salário médio.

A consulta externa pega o resultado da subconsulta e retorna combinações que são maiores.

A conexão natural nos dá os nomes das pessoas com salário maior que aquele retornado pela consulta interna.

```
SELECT mc.primeiro_nome, mc.sobrenome
FROM meus_contatos mc
NATURAL JOIN emprego_atual ea
WHERE ea.salario > (SELECT AVG(salario) FROM emprego_atual);
```

A subconsulta retorna o salário médio.

Encontre todos os Web Designers que possuem o mesmo CEP que aqueles da tabela lista\_emploies para Web Designers.

Precisamos utilizar uma conexão natural para obter informações úteis, como nomes e número de telefone para as pessoas que encontrarmos.

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone
FROM meus_contatos mc
NATURAL JOIN emprego_atual ea
WHERE ea.cargo = 'Programador web' AND mc.cep
IN (SELECT cep FROM lista_emploies WHERE cargo = 'Programador web');
```

Devido ao fato que poderia existir mais ceps retornados, tratamos os resultados como um conjunto e utilizando a palavra-chave `IN` para encontrar a combinação.

A consulta interna retorna todos os ceps para as vagas de web designer da lista de empregos.

Liste qualquer pessoa que viva no mesmo CEP que a pessoa que atualmente possui o maior salário.

Esta é uma pergunta complicada porque poderia haver mais de uma pessoa com o maior salário. Isto significa que precisaremos utilizar uma `IN`. Nós também precisaremos usar duas subconsultas.

A consulta externa pega o cep e encontra compatíveis na tabela `meus_contatos` porque a subconsulta do meio poderia ter retornado mais que um CEP, então utilizamos `IN`.

A consulta do meio encontrou CEPs de pessoas que recebem o maior salário.

```
SELECT sobrenome, primeiro_nome
FROM meus_contatos
WHERE c_cep IN (SELECT mc.cep
FROM meus_contatos mc
NATURAL JOIN emprego_atual ea
WHERE ea.salario = (SELECT MAX(salario) FROM emprego_atual));
```

A subconsulta mais profunda obteve o valor `MAX` do salário da tabela `emprego_atual`. Isto será um valor único, então podemos utilizar `=`.

## Subconsultas correlacionadas



Se uma subconsulta não-correlacionada significa dizer que a subconsulta funciona sozinha, então eu aposto que uma consulta correlacionada é de alguma forma dependente da consulta externa.

**Correto. Em uma subconsulta não-correlacionada, a consulta interna, ou subconsulta, é interpretada pelo Sistema SQL, seguida pela consulta externa.**

O que nos deixa com a subconsulta correlacionada. Uma subconsulta correlacionada quer dizer que a consulta interna depende da consulta externa antes de poder ser resolvida.

A consulta abaixo conta o número de interesses na tabela `interesses` para cada pessoa em `meus_contatos`, então retorna o primeiro nome e o sobrenome daqueles que têm três interesses.

```
SELECT mc.primeiro_nome, mc.sobrenome
FROM meus_contatos AS mc
WHERE
  3 = (
    SELECT COUNT(*) FROM contato_
    interesse
    WHERE id_contato = mc.id_contato
  );
```

O pseudônimo para `meus_contatos` é criado na consulta externa.

A subconsulta refere-se ao pseudônimo `mc`.

A consulta externa tem que ser executada antes de sabermos qual é o valor de `mc.id_contato`.

A subconsulta depende da consulta externa. Ela precisa do valor de `id_contato` da consulta externa antes da consulta interna ser processada.

Ela usa o mesmo pseudônimo, ou nome correlacionado, para `meus_contatos`, `mc`, que foi criado na consulta externa.

## Uma subconsulta correlacionada (útil) com NOT EXISTS

Um uso bastante comum para subconsultas correlacionadas é encontrar todas as linhas de uma consulta externa para qual não exista nenhuma linha em uma tabela relacionada.

Suponha que Greg precise de mais clientes para seu crescente negócio de recrutamento e quer mandar um email para todos na tabela `meus_contatos` que atualmente **não estão** na tabela `emprego_atual`. Ele pode utilizar um `NOT EXISTS` para focalizar estas pessoas.

```
SELECT mc.primeiro_nome primeironome, mc.sobrenome sobrenome, mc.email email
FROM meus_contatos mc
WHERE NOT EXISTS
  (SELECT * FROM emprego_atual ea
  WHERE mc.id_contato = ea.id_contato);
```

`NOT EXISTS` encontra o primeiro nome, sobrenome e email das pessoas da tabela `meus_contatos` que ainda não estão atualmente listadas na tabela `emprego_atual`.

## QUAL O MEU PROPÓSITO

Relacione cada parte da consulta com o que ela faz

<code>mc.primeiro_nome primeironome</code>	Define um pseudônimo para o campo mc_sobrenome
<code>WHERE NOT EXISTS</code>	Se dois id_contatos são verdadeiros, a condição é encontrada
<code>WHERE mc.id_contato = ea.id_contato</code>	Define um campo "primeironome" como um pseudônimo
<code>FROM meus_contatos mc</code>	Seleciona todos os campos para a tabela com o pseudônimo "ea"
<code>mc.sobrenome sobrenome</code>	Define um campo "email" como um pseudônimo
<code>SELECT * FROM emprego_atual ea</code>	Especifica valor verdadeiro se algo não for encontrado
<code>mc.email email</code>	Define um pseudônimo para meus_contatos

## EXISTS e NOT EXISTS

Da mesma forma como com IN e NOT IN, você pode utilizar ambos **EXISTS** e NOT EXISTS com suas subconsultas. A consulta abaixo retorna dados de meus\_contatos onde o id\_contato aparece pelo menos uma vez na tabela contato\_interesse.

```
SELECT mc.primeiro_nome primeironome, mc.sobrenome sobrenome, mc.email email
FROM meus_contatos mc
WHERE EXISTS ← EXISTS encontra o primeiro nome, o sobrenome e o email das pessoas na tabela meus_
contatos, nas quais o id_contato aparece pelo menos uma vez na tabela contato_interesse.
(SELECT * FROM contato_interesse ci WHERE mc.id_contato = ci.id_contato);
```

## QUAL O MEU PROPÓSITO

Relacione cada parte da consulta com o que ela faz

<code>mc.primeiro_nome primeironome</code>	Define um pseudônimo para o campo mc_sobrenome
<code>WHERE NOT EXISTS</code>	Se dois id_contatos são verdadeiros, a condição é encontrada
<code>WHERE mc.id_contato = ea.id_contato</code>	Define um campo "primeironome" como um pseudônimo
<code>FROM meus_contatos mc</code>	Seleciona todos os campos para a tabela com o pseudônimo "ea"
<code>mc.sobrenome sobrenome</code>	Define um campo "email" como um pseudônimo
<code>SELECT * FROM emprego_atual ea</code>	Especifica valor verdadeiro se algo não for encontrado
<code>mc.email email</code>	Define um pseudônimo para meus_contatos



## Aponte seu lápis

Escreva uma consulta que retorne o email das pessoas que tenham, ao menos, um interesse, mas que não se encontram na tabela emprego\_atual.

→ Resposta na página 338.

## A empresa de Greg para serviços de recrutamento está aberta para negócios

Greg está agora confortável em obter os dados com suas subconsultas. Ele até descobriu que pode utilizar os comandos INSERT, UPDATE e DELETE.

Ele alugou um pequeno espaço para escritório para sua nova empresa, e decide ter uma festa de inauguração.



Eu imagino se posso encontrar meu próprio e primeiro funcionário na tabela emprego\_desejado

## não existem Perguntas Idiotas

P: Então você pode colocar uma subconsulta dentro de uma subconsulta?

R: Definitivamente. Há um limite em quantas subconsultas aninhadas você pode usar, mas a maioria dos Sistemas SQL tolera bem mais que a capacidade que possa conseguir usar.

P: Qual a melhor abordagem quando se está tentando construir uma subconsulta dentro de uma subconsulta?

R: Seu melhor palpite seria escrever pequenas consultas para as partes da questão. Depois olhe para elas e observe como você precisa combiná-las. Se está tentando encontrar pessoas que ganham a mesma quantidade de salário que aquele que percebe o maior salário pago a um web designer, divida-as em:

Encontrar o maior salário para um web designer  
Encontrar pessoas que ganham x quantidade do salário

Após coloque a primeira resposta no lugar do x.

P: Se eu não gosto de utilizar subconsultas, há alguma possibilidade de usar conexões no lugar.

R: Na maioria das vezes, sim. Você precisa, entretanto. O que nos leva a...

## o caminho da festa

Coloca esse tablóide perturbador na capa:

# O CONSULTOR SEMANAL

## A VERDADE BOMBÁSTICA sobre subconsultas REVELADO!

### Conexões escondidas

Vizinhos disseram que as subconsultas não podem fazer "nada mais" que as conexões, e "a verdade precisa prevalecer".

**Por Troy Armstrong**

Equipe de Escritores de O Consultor

DATAVILLE - O que tem sido apenas especulação por muitos anos foi agora verificado pelas fontes de O Consultor. Subconsultas e Conexões podem ser utilizadas para fazerem exatamente as mesmas consultas. Por mais confuso que seja para os residentes locais, tudo o que pode fazer com uma subconsulta, pode-se fazer com algum tipo de conexão.

"É terrível", lamentou a professora Heidi Musgrove. "Como posso dizer às crianças que o que elas pensavam saber sobre as subconsultas, todas aquelas horas aprendendo como utilizá-las, bem, elas poderia ter apenas utilizado conexões. É de partir o coração".

O desfecho desta revelação é esperado para o próximo capítulo, quando as consultas externas são reveladas para o público.



Residente local Heidi Musgrove estava chocada em saber a verdade sobre as subconsultas.

ESTO TUDO FOI UMA PERDA DE TEMPO? AS SUBCONSULTAS SÃO  
REALMENTE A MESMA COISA QUE AS CONEXÕES? VÁ PARA O PRÓXIMO  
CAPÍTULO PARA DESCOBRIR.

alário

alguma

que nos



## Sua caixa de ferramentas SQL

Você acabou de completar o Capítulo 9 e já é mestre na arte da subconsulta. Dê uma olhada no que aprendeu. Para uma lista completa de dicas neste livro, veja o Apêndice iii.

**Subconsulta não-correlacionada**  
Uma subconsulta que funciona sozinha e não faz referência a uma consulta externa.

**Subconsulta correlacionada**  
Uma subconsulta que depende dos valores retornados pela consulta externa.

**Consulta externa**  
Uma consulta que contém uma consulta interna ou subconsulta.

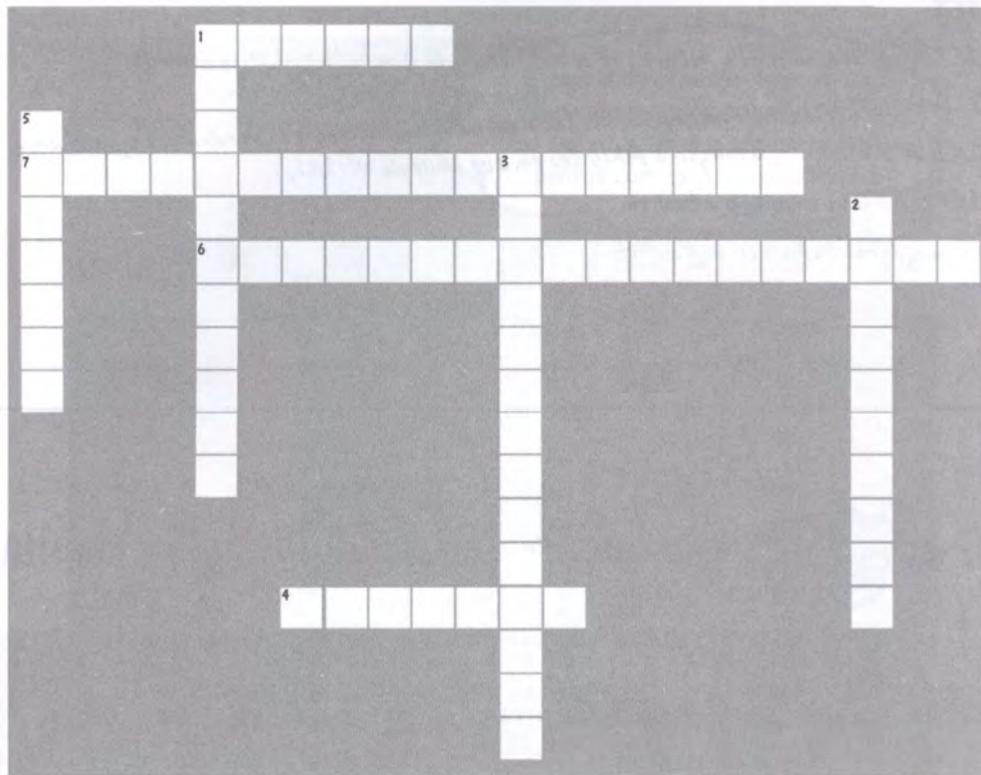
**Consulta interna**  
Uma consulta dentro de outra consulta. É também conhecida como subconsulta.

**Subconsulta**  
Uma consulta embrulhada dentro de outra consulta. É também conhecida como consulta interna.



## Cruzada-Subconsulta

Você pode diferenciar sua consulta interna da sua consulta externa, mas será que você pode resolver este passatempo? Todas as respostas são deste capítulo.



### Horizontais

1. Uma subconsulta é sempre um só comando \_\_\_\_\_
  4. As consultas \_\_\_\_\_ contêm a consulta interna ou subconsulta.
  6. Se a subconsulta funciona sozinha e não faz referência a nenhuma outra.
  7. Em uma subconsulta \_\_\_\_\_, a consulta interna ou subconsulta é interpretada pelo Sistema SQL seguida da consulta interna.

## Verticais

1. Uma consulta dentro de outra consulta é chamada de \_\_\_\_\_.
  2. Subconsultas estão sempre dentro de uma \_\_\_\_\_.
  3. Uma consulta \_\_\_\_\_ quer dizer que a consulta interna depende da consulta externa antes de ser resolvida.
  5. A consulta \_\_\_\_\_, é chamada de subconsulta.

Resposta na próxima página



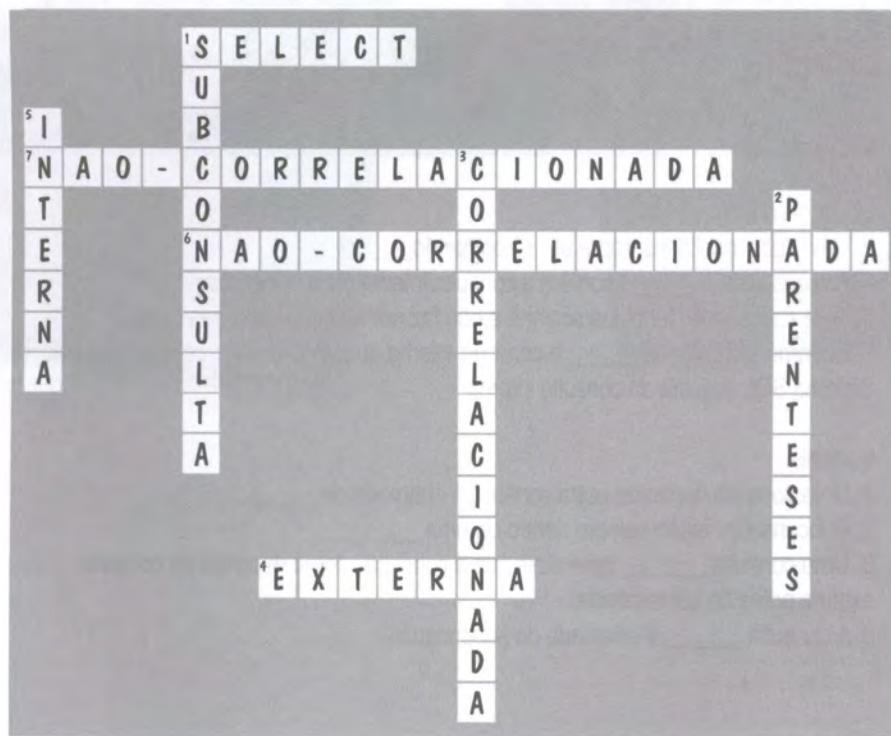
Aponte seu lápis  
Solução  
Da página 334

Escreva uma consulta que retorne o email das pessoas que tenham, ao menos, um interesse, mas que não se encontram na tabela emprego\_atual.

```
SELECT mc.email FROM meus_contatos mc WHERE
EXISTS
(SELECT * FROM contato_interesse ci WHERE mc.id_contato = ci.id_contato)
AND ← Da mesma forma como quando duas coisas são verdadeiras, você deve
NOT EXISTS utilizar o AND (e) em sua cláusula WHERE.
(SELECT * FROM emprego_atual ea
WHERE mc.id_contato = ea.id_contato);
```



## Solução da Cruzada-Subconsulta



## 10 Conexões externas, intraconexões e uniões

### Novas manobras



**Você só sabe metade da história sobre conexões.** Você já viu conexões cruzadas que retornam todas as linhas possíveis e consultas internas que retornam valores de ambas as tabelas onde há uma combinação. Mas o que ainda não as consultas externas que retornam as linhas que não possuem duplicatas compatíveis na outra tabela, intraconexão que (por mais estranho que pareça) conecta uma só tabela a ela mesma, e uniões que combinam resultados de consultas. Uma vez que tenha aprendido esses truques, você será capaz de acessar todos os seus dados exatamente da forma que precisar (e nós não esquecemos sobre expor a verdade sobre subconsultas!)

## Limpando os dados antigos



Eu gostaria de limpar minha tabelas de profissões. Acho que posso ter alguns valores ali que eu não estou mais utilizando. Como posso facilmente encontrar profissões que não estão conectadas a nenhum dos registros na tabela meus\_contatos? Não posso criar uma consulta interna só para fazer isso.

### Você pode obter aquela informação com uma consulta externa

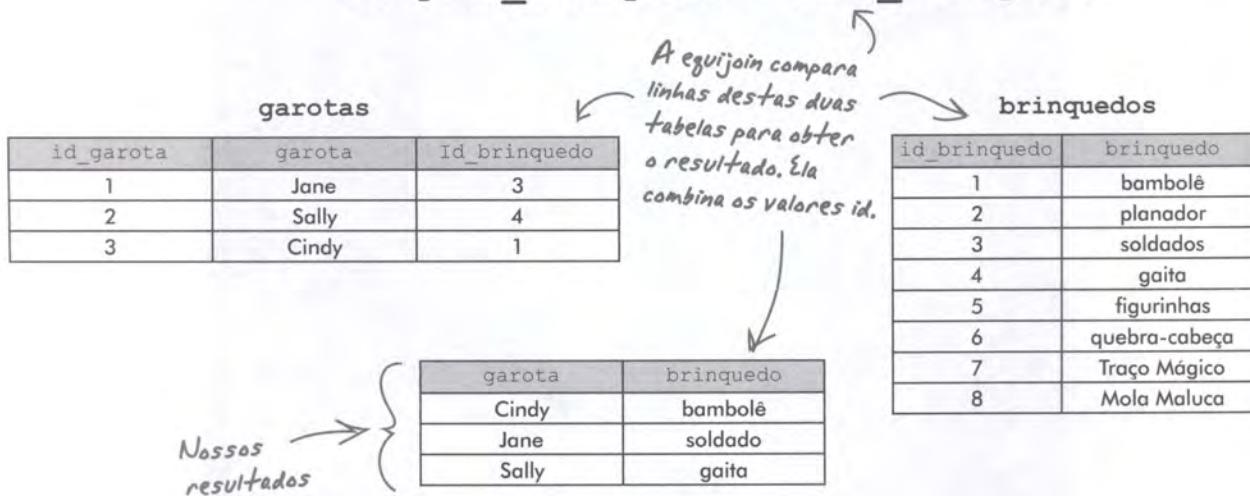
Vamos dar uma olhada no que as consultas externas fazem, então ensinaremos como encontrar aquelas profissões que você não está mais utilizando.

Uma consulta externa retorna todas as linhas de uma das tabelas bem como as informações compatíveis de outra tabela.

Com uma consulta interna, você está **comparando linhas de duas tabelas, mas a ordem destas duas tabelas não importa**.

Vamos rever brevemente o que a equijoin faz. Pegamos todas as colunas que são compatíveis com `id_brinquedo` de ambas as tabelas. Ela combina as duas colunas `id_brinquedo` que existem em ambas as tabelas.

```
SELECT g.garota, b.brinquedo
FROM garotas g
INNER JOIN brinquedos b
ON g.id_brinquedo = b.id_brinquedo;
```



## É uma questão de esquerda e direita

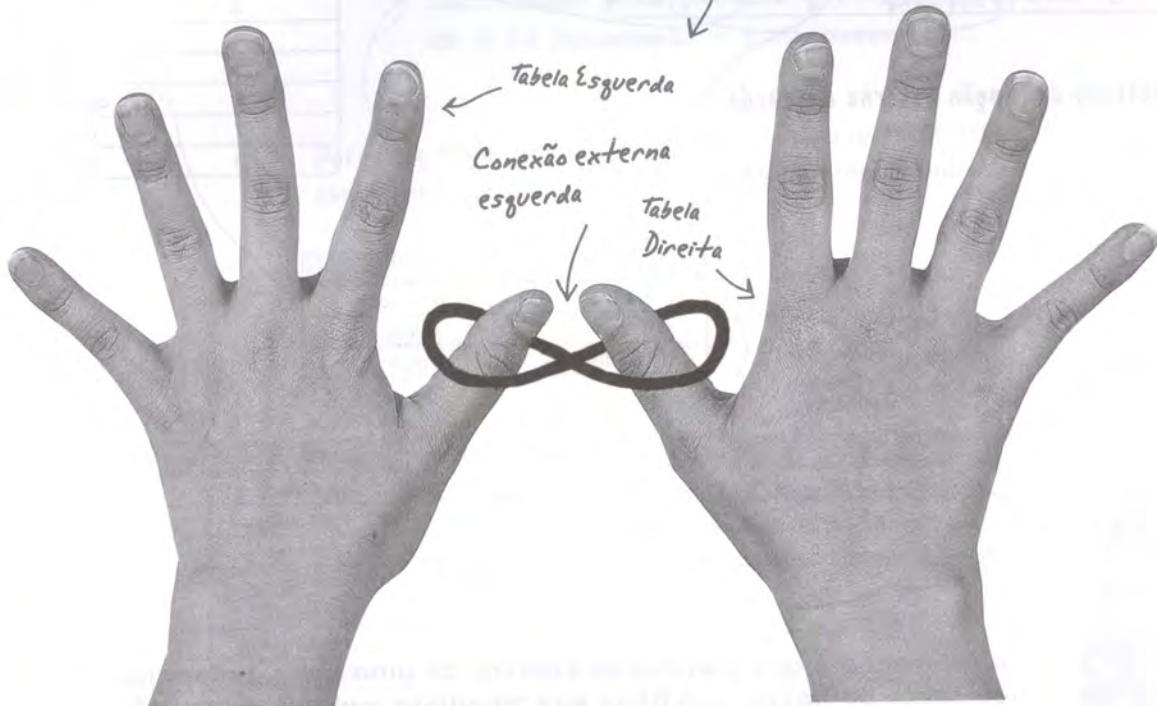
Em comparação, conexões externas têm mais a ver com **relacionamento entre duas tabelas** que todas as outras conexões que já vimos.

Uma Conexão Externa ESQUERDA pega todas as linhas da tabela esquerda e combina com as linhas da tabela DIREITA. É útil quando a tabela esquerda e a tabela direita possuem um relacionamento um-para-muitos.

A conexão externa esquerda combina **CADA LINHA** na tabela **ESQUERDA** com uma linha na tabela direita.

grande segredo de entender as conexões externas é saber qual tabela está à esquerda e qual está à direita.

A CONEXÃO EXTERNA ESQUERDA, a tabela que vem depois de **FROM** e **ANTES** da conexão é uma tabela **ESQUERDA**, e a tabela que vem depois da conexão é uma **TABELA DIREITA**.



## Aqui está uma conexão externa esquerda

Vamos utilizar uma conexão externa esquerda para descobrir qual garota tem qual brinquedo.

Aqui está a sintaxe de uma conexão externa esquerda utilizando as mesmas tabelas de antes. A tabela garotas é a primeira depois de **FROM** e, finalmente, a tabela brinquedos é a tabela da **DIREITA**:

Então, a **CONSULTA EXTERNA ESQUERDA** pega todas as linhas na tabela esquerda (a tabela garotas) e combina suas linhas com a tabela DIREITA (a tabela brinquedos).

```
SELECT g.garota, b.brinquedo
```

```
FROM garotas g
```

```
LEFT OUTER JOIN brinquedos b
```

```
ON g.id_brinquedo = b.id_brinquedo;
```

Ela vem antes da conexão externa esquerda, então garotas é a tabela esquerda...

...e porque esta tabela vem depois da conexão externa direita, brinquedos é a tabela direita.

Esta vem antes da conexão externa esquerda, então garotas é a tabela esquerda...

...e porque esta tabela vem depois de conexão externa direita, brinquedos é a tabela direita.

id_garota	garota	Id_brinquedo
1	Jane	3
2	Sally	4
3	Cindy	1

id_brinquedo	brinquedo
1	bambolê
2	planador
3	soldados
4	gaita
5	figurinhas
6	quebra-cabeça
7	Traço Mágico
8	Mola Maluca

### O resultado da junção externa esquerda

Os resultados são os mesmos que os resultados da conexão interna.

garota	brinquedo
Cindy	bambolê
Jane	soldado
Sally	gaita

Nossos resultados

Então é isto? Qual é o grande problema, então? A conexão externa parece ser a mesma coisa que a conexão interna.



**A diferença é que a conexão externa dá uma linha, independentemente de haver uma linha que se ajuste a ela na outra tabela.**

E um valor NULL diz que não existe uma combinação. No caso de nossas tabelas garotas e brinquedos, um valor NULL nos resultados significa que um brinquedo, em particular, não pertence a nenhuma das garotas. Esta é uma informação valiosa!

Um valor NULL nos resultados de uma conexão externa esquerda quer dizer que a tabela direita não possui nenhum valor correspondente à tabela esquerda.



Aponte seu lápis —

Rascunhe o que você acha que será o resultado desta tabela.

```
SELECT g.garota, b.brinquedo
FROM brinquedos b
LEFT OUTER JOIN garotas g
ON b.id_brinquedo = g.id_brinquedo;
```

(Dica: Haverá 8 linhas na tabela resultado.)


**Aponte seu lápis**  
**Solução**

Rascunhe o que você acha que será o resultado desta tabela.

```
SELECT g.garota, b.brinquedo
FROM brinquedos b ← A tabela esquerda
LEFT OUTER JOIN garotas g ← A tabela direita.
ON b.id_brinquedo = g.id_brinquedo;
```

Desta vez, todas as linhas na tabela brinquedos (a tabela esquerda) são comparadas com a tabela garotas (a tabela direita).

id_brinquedo	brinquedo
1	bambolê
2	planador
3	soldados
4	gaita
5	figurinhas
6	quebra-cabeça
7	Traço Mágico
8	Mola Maluca

id_garota	garota	Id_brinquedo
1	Jane	3
2	Sally	4
3	Cindy	1

Com a ordem de nossa tabela alterada, isso é o que obtemos.

garota	brinquedo
Cindy	bambolê
NULL	planador
Jane	soldado
Sally	gaita
NULL	figurinhas
NULL	quebra-cabeça
NULL	Traço Mágico
NULL	Mola Maluca

Se uma combinação é encontrada, ela é exibida como um resultado em nossa tabela, nós ainda teremos um valor em nossa tabela, mas do tipo NULL para os valores sem combinações.

A ordem que as colunas aparecem na tabela é a ordem na qual nós as selecionamos. Esta ordem não tem nada a ver com a conexão ESQUERDA.



A consulta

Abaixo estão dois conjuntos de resultados. Para cada conjunto de resultado escreva uma conexão externa esquerda que poderia tê-lo criado, junto com as tabelas garotas e brinquedos com dados que combinam com os resultados.

Resultado de uma conexão externa esquerda:

garota	brinquedo
Jen	Pistola de água
Cleo	Canudos malucos
Mandy	NULL

Tabela Esquerda

Nós fizemos esta  
aqui para você.  
↙  
garotas

Tabela Direita

<i>id_garota</i>	<i>garota</i>	<i>id_bringuedo</i>
1	Jen	1
2	Cleo	2
3	Mandy	3

A consulta

Resultado da nossa  
conexão externa esquerda:

Está aqui é  
complicada. ↘

garota	brinquedo
Jen	Pistola de água
Cleo	Pistola de água
NULL	Canudos malucos
Sally	Mola Maluca
Martha	Mola Maluca

Tabela Esquerda

Tabela Direita



## Solução dos Exercícios

A consulta

```
SELECT g.garota, b.brinquedo
FROM garotas g
LEFT OUTER JOIN brinquedos b
ON g.id_brinquedo = b.id_brinquedo;
```

Tabela Esquerda

*garotas*

<i>id_garota</i>	<i>garota</i>	<i>id_brinquedo</i>
1	Jen	1
2	Cleo	2
3	Mandy	3

Este pode ser qualquer *id\_brinquedo* que não existe na tabela *brinquedos* desde que a coluna *brinquedo*

A consulta acaba com resultados NULL.

Resultado de uma conexão externa esquerda:

<i>garota</i>	<i>brinquedo</i>
Jen	Pistola de água
Cleo	Canudos malucos
Mandy	NULL

Estes são os brinquedos exibidos em nossos resultados.

Tabela Direita

*brinquedos*

<i>id_brinquedo</i>	<i>brinquedo</i>
1	Pistola de Água
2	Canudos malucos

Os valores repetidos significam que mais de uma garota tem o mesmo brinquedo.

Resultado da nossa conexão externa esquerda:

<i>garota</i>	<i>brinquedo</i>
Jen	Pistola de água
Cleo	Pistola de água
NULL	Canudos malucos
Sally	Mola Maluca
Martha	Mola Maluca

É o valor NULL quer dizer que nenhuma garota tem os canudos malucos.

Tabela Esquerda

*brinquedos*

1	Pistola de água
2	Canudos malucos
3	Mola Maluca
3	slinky

Tabela Direita

*garotas*

<i>id_garota</i>	<i>garota</i>	<i>id_brinquedo</i>
1	Jen	1
2	Cleo	1
3	Sally	3
4	Martha	3

## Conexões externas e combinações múltiplas

Como percebeu no exercício, você obterá linhas mesmo que não haja combinações na outra tabela, bem como as linhas múltiplas quando existirem múltiplas combinações. Aqui está o que a conexão externa esquerda realmente está fazendo:

```
SELECT g.garota, b.brinquedo
FROM brinquedos b
LEFT OUTER JOIN garotas g
ON g.id_brinquedo = b.id_brinquedo;
```

brinquedos		garotas		
id_brinquedo	brinquedo	id_garota	garota	id_brinquedo
1	Pistola de água	1	Jen	1
2	Canudos malucos	2	Cleo	1
3	Molas Malucas	3	Sally	3
		4	Martha	3

A linha da tabela brinquedos, Pistola de água é comparada com a Linha Jen da tabela garotas: brinquedos.  
id\_brinquedo = 1, garotas.id\_brinquedo = 1

**Nós temos uma combinação.**

A linha da tabela brinquedos, Pistola de água é comparada com a Linha Cleo da tabela garotas: brinquedos.  
id\_brinquedo = 1, garotas.id\_brinquedo = 1

**Nós temos uma combinação.**

A linha da tabela brinquedos, Pistola de água é comparada com a Linha Sally da tabela garotas: brinquedos.  
id\_brinquedo = 1, garotas.id\_brinquedo = 3

**Não há combinação.**

A linha da tabela brinquedos, Pistola de água é comparada com a Linha Martha da tabela garotas: brinquedos.  
id\_brinquedo = 1, garotas.id\_brinquedo = 3

**Não há combinação.**

A linha da tabela brinquedos, Canudos malucos é comparada com a Linha Jen da tabela garotas: brinquedos.  
id\_brinquedo = 2, garotas.id\_brinquedo = 1

**Não há combinação.**

A linha da tabela brinquedos, Canudos malucos é comparada com a Linha Cleo da tabela garotas: brinquedos.  
id\_brinquedo = 2, garotas.id\_brinquedo = 1

**Não há combinação.**

A linha da tabela brinquedos, Canudos malucos é comparada com a Linha Sally da tabela garotas:  
brinquedos.id\_brinquedo = 2, garotas.id\_brinquedo = 3

**Não há combinação.**

A linha da tabela brinquedos, Canudos malucos é comparada com a Linha Martha da tabela garotas:  
brinquedos.id\_brinquedo = 2, garotas.id\_brinquedo = 3

**Não há combinação.**

**Final da tabela, uma linha com NULL é criada.**

A linha da tabela brinquedos, Molas Malucas é comparada com a Linha Jen da tabela garotas: brinquedos.  
id\_brinquedo = 3, garotas.id\_brinquedo = 1

**Não há combinação.**

A linha da tabela brinquedos, Molas Malucas é comparada com a Linha Cleo da tabela garotas: brinquedos.  
id\_brinquedo = 3, garotas.id\_brinquedo = 1

**Não há combinação.**

A linha da tabela brinquedos, Molas Malucas é comparada com a Linha Sally da tabela garotas: brinquedos.  
id\_brinquedo = 3, garotas.id\_brinquedo = 3

**Nós temos uma combinação.**

A linha da tabela brinquedos, Molas Malucas é comparada com a Linha Martha da tabela garotas: brinquedos.  
id\_brinquedo = 3, garotas.id\_brinquedo = 3

**Nós temos uma combinação.**

garota	brinquedo
Jen	Pistola de água
Cleo	Pistola de água
NULL	Canudos malucos
Sally	Molas Malucas
Martha	slinky

## Coneção externa direita

A conexão externa direita é a mesma coisa que conexão externa esquerda, exceto quando compara a tabela direita com a esquerda. As duas consultas abaixo darão precisamente os mesmos resultados:

```
SELECT g.garota, b.brinquedo
FROM brinquedos b ← A tabela direita.
LEFT OUTER JOIN garotas g ← esquerda.
ON g.id_brinquedo = b.id_brinquedo;
```

A tabela esquerda  
(em ambas as consultas)

Id_garota	garota	Id_brinquedo
1	Jane	3
2	Sally	4
3	Cindy	1

As duas consultas fazem da tabela garotas a tabela esquerda.

Nossos resultados

garota	brinquedo
Cindy	bambolê
Jane	soldado
Sally	gaita

Tabela direita  
Coneção externa direita  
Tabela esquerda

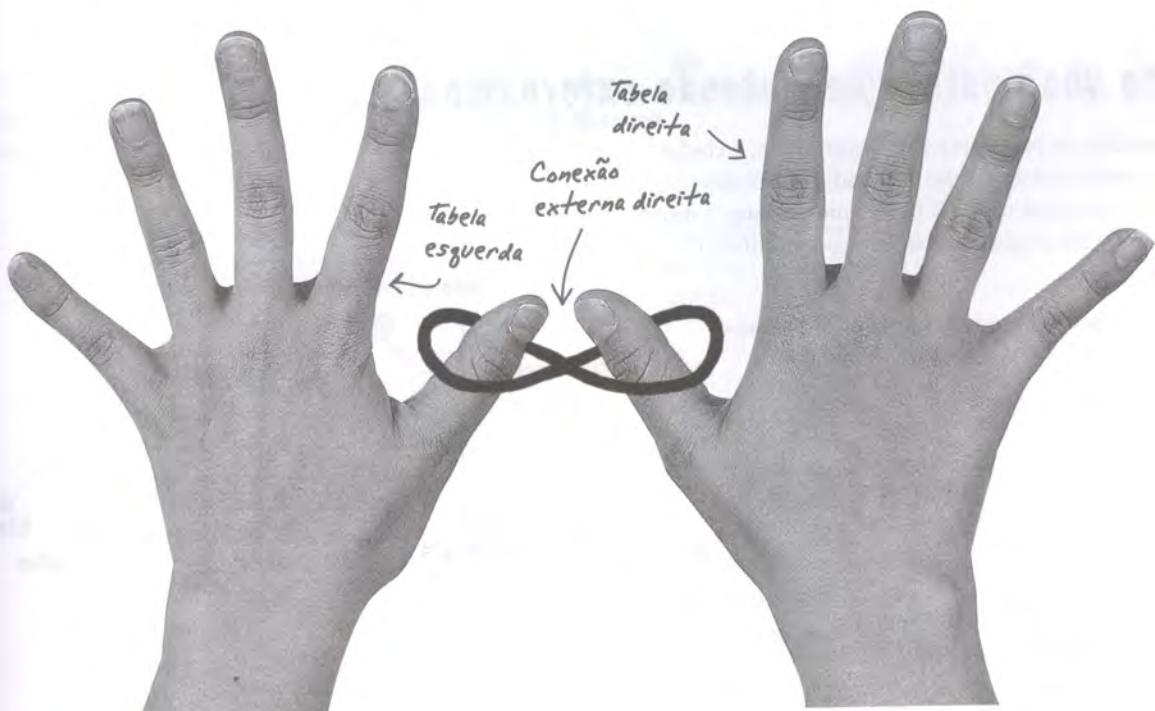
A conexão externa direita avalia a tabela direita em contraposição à tabela esquerda.

```
SELECT g.garota, b.brinquedo
FROM garotas g ← A tabela esquerda. A tabela
LEFT OUTER JOIN brinquedos b ← direita.
ON g.id_brinquedo = b.id_brinquedo;
```

Você já viu esta consulta na página 342

A tabela direita (em ambas as consultas)

id_brinquedo	brinquedo
1	bambolê
2	planador
3	soldados
4	gaita
5	figurinhas
6	quebra-cabeça
7	Traço Mágico
8	Mola Maluca



## Perguntas Ídiotas

**P:** Há alguma razão para utilizar a conexão externa esquerda ao invés da direita?

**R:** Alterar a palavra ESQUERDA (LEFT) para direita (RIGHT) é mais fácil que alterar a ordem das tabelas em sua consulta. Você só tem que alterar uma palavra, ao invés de trocar o nome das duas tabelas e seus pseudônimos.

Em geral, é na verdade mais fácil se ater a um só tipo, digamos a esquerda, e alterar qual tabela é esquerda ou direita. Isto pode ser menos confuso.

**P:** Então se há uma conexão externa ESQUERDA e uma conexão externa DIREITA, existe alguma conexão que retorne ambos os resultados das conexões esquerda e direita?

**R:** Existe em alguns, mas não todos os Sistemas SQL e é chamado de FULL OUTER JOIN, mas não funciona com MySQL, SQL Server ou Access.



Você não poderia usar uma conexão externa para unir uma tabela simples nela mesma? Deve ser bastante útil.

### **Você pode utilizar a mesma tabela tanto como esquerda ou direita na conexão externa.**

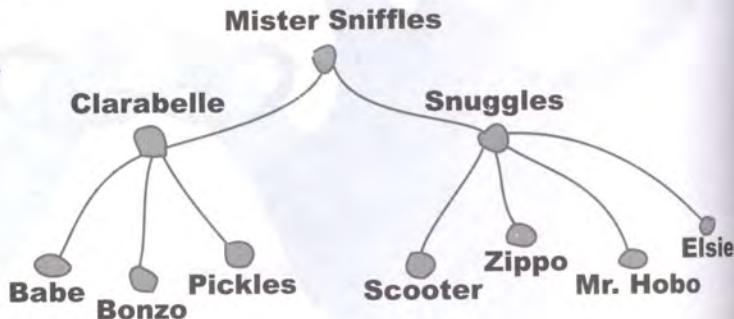
E enquanto isso parece estranho, isso é muito útil. Dê uma olhada em uma situação quando você poderá precisar da conexão externa propriamente dita, para uma tabela.

Primeiro, há um grande problema com os palhaços em Dataville.

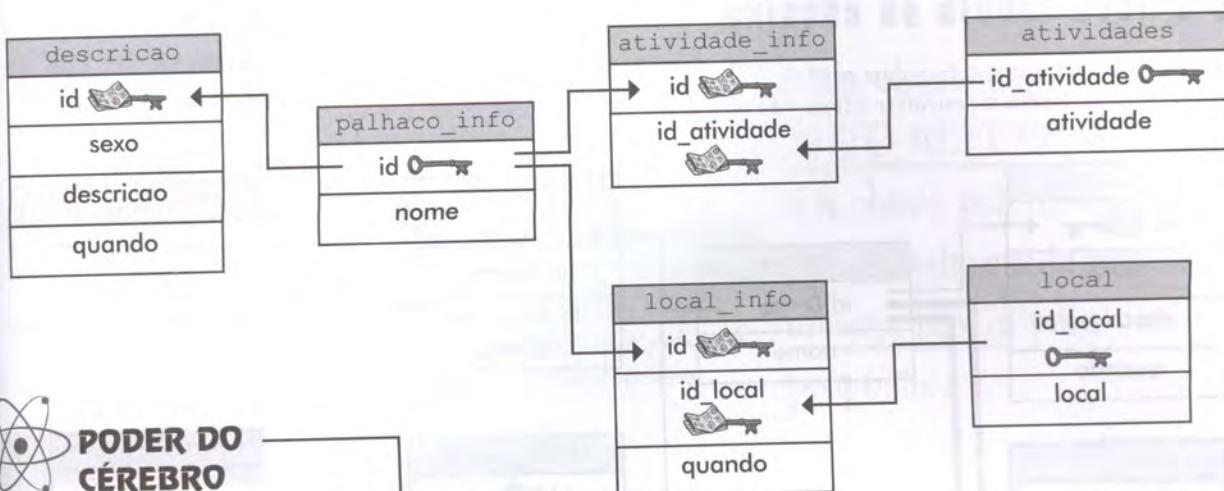
## **Enquanto você estava conectando externamente...**

De volta a Dataville, os palhaços estão organizando, e chefes palhaços estão sendo colocados no comando. Este é um desenvolvimento assustador, e nós precisamos manter o rastro de quem são os chefes, e quais palhaços seguem ordem de quais chefes.

Aqui está um exemplo da nova hierarquia dos palhaços. Cada palhaço tem um chefe, exceto pelo palhaço principal, Mister Sniffles.



Vamos dar uma olhada no nosso esquema atual e ver como melhor encaixar esta informação:



### PODER DO CÉREBRO

Como você pode reestruturar seu esquema para armazena as informações sobre os palhaços chefes?

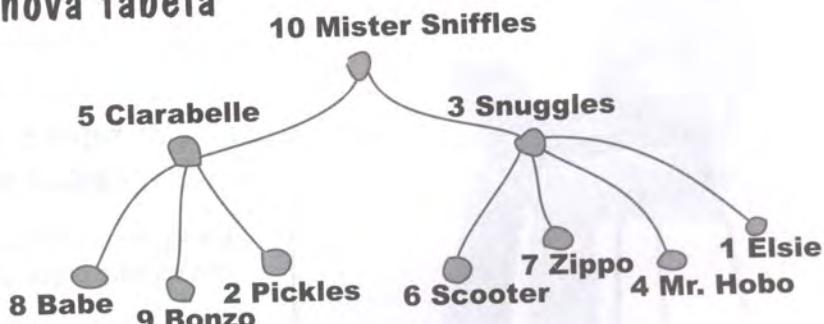


Bom, como  
sou engraçado? Quero  
dizer, engraçado como  
um palhaço, eu te  
entretenho?

## Nós poderíamos criar uma nova tabela

Nós podemos criar uma tabela que relate cada palhaço e ID de seu respectivo chefe. Aqui está essa hierarquia com o ID de cada palhaço.

Aqui está a nova tabela que relaciona cada palhaço e o id do respectivo chefe na tabela z.



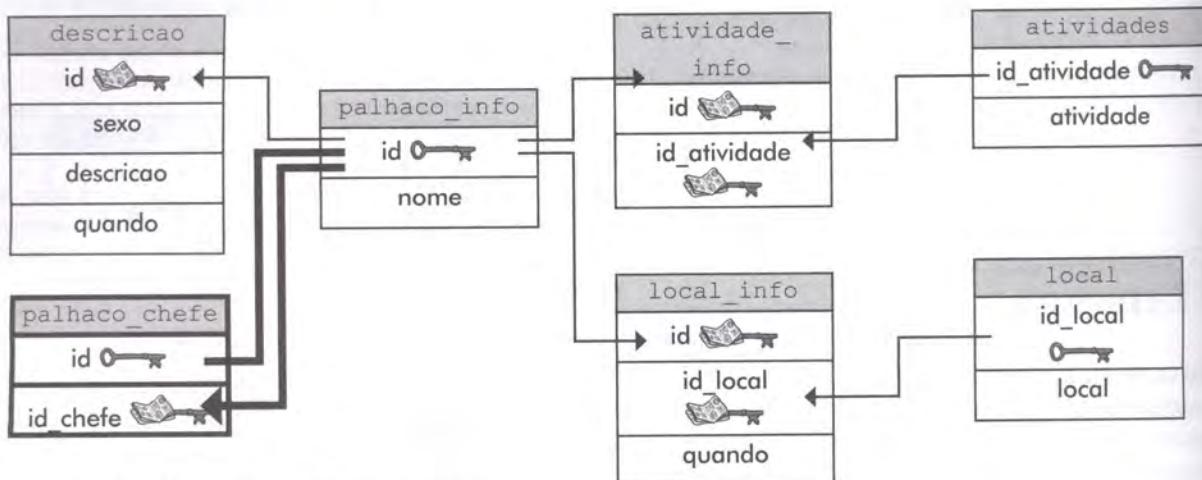
palhaco_chefe	
palhaco	id_chefe
1	3
2	5
3	10
4	3
5	10
6	3
7	3
8	5
9	5
10	10

Temos um relacionamento um-a-um entre as tabelas `palhaco_chefe` e `palhaco_info`.

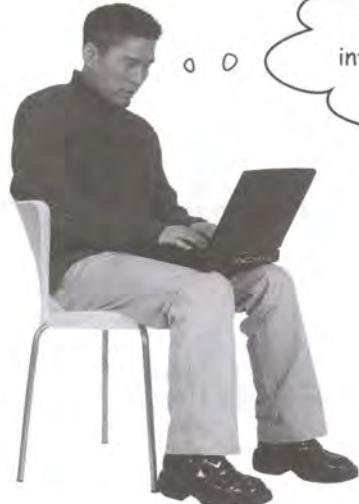
Mister Sniffles não tem chefe algum, mas ele precisa de um id. Nós podemos colocar seu próprio id na coluna correspondente para evitar que exista NULL nessa coluna.

## Como a nova tabela se encaixa

Vamos dar uma olhada no nosso esquema atual e verificar qual a melhor forma de encaixar a nova tabela:



É um pouco estranho. Temos um relacionamento um-a-um com id – nossa chave-primária – e relacionamento um-para-muitos com nossa tabela id\_chefe. Temos uma chave-primária e uma chave estrangeira **ambas da tabela palhaco\_info**.



Parece que você poderia utilizar uma tabela um-para-um, já que não há nenhuma informação particular ali, não dá para encaixar isto na tabela principal de alguma forma?

### PODER DO CÉREBRO

Há algum jeito de podermos rastrear os palhaços chefes sem precisar criar uma nova tabela?

## Uma chave estrangeira auto-referenciada

O que precisamos é de uma nova coluna em nossa tabela palhaco\_info que nos diga quem é o chefe de cada palhaço. A nova coluna conterá o número ID do chefe de cada palhaço. Nós a chamaremos de `id_chefe`, da mesma forma que na tabela palhaco\_chefe.

Na tabela palhaco\_chefe, `id_chefe` é uma chave estrangeira. Quando adicionamos a coluna para palhaco\_info, ela continua sendo uma chave estrangeira, ainda que esteja na tabela palhaco\_info. Isto é conhecido como **uma chave estrangeira auto-referenciada**. A parte auto-referenciada significa que ela é uma chave que está fazendo referência a outro campo na mesma tabela.

Presumimos que Mister Sniffles é seu próprio chefe, então seu `id_chefe` é o mesmo que seu `id`. Isto quer dizer que podemos utilizar uma chave estrangeira auto-referenciada como nosso `id_chefe`.

**Uma chave estrangeira auto-referenciada** é a **chave primária** de uma tabela **usada naquela mesma tabela para outro propósito**.

é a nova coluna `id_chefe` que simplesmente adicionamos à coluna `palhaco_info`. Ela contém uma chave primária auto-referenciada.

`palhaco_info`

id	nome	id_chefe
1	Elsie	3
2	Pickles	5
3	Snuggles	10
4	Mr. Hobo	3
5	Clarabelle	10
6	Scooter	3
7	Zippo	3
8	Babe	5
9	Bonzo	5
10	Mister Sniffles	10

Ela faz referência ao campo `id` na mesma tabela para nos dizer qual palhaço é o chefe de Elsie.

Mais uma vez, o `id_chefe` de Mister Sniffles é o seu próprio.

## Uma chave estrangeira AUTO-REFERENCIADA

é a chave primária de uma tabela usada naquela mesma tabela para outro propósito.

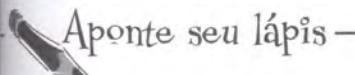
## Conecte uma tabela a ela mesma

Suponha que queiramos listar cada palhaço e quem é o chefe de cada um deles. Podemos facilmente obter uma lista com o nome de cada palhaço e o id de seus chefes com este SELECT:

```
SELECT nome, id_chefe FROM palhaco_info;
```

O que realmente queremos é o nome do palhaço e o nome do chefe dele palhaço.

Nome	chefe
Elsie	Snuggles
Pickles	Clarabelle
Snuggles	Mister Sniffles
Mr. Hobo	Snuggles
Clarabelle	Mister Sniffles
Scooter	Snuggles
Zippo	Snuggles
Babe	Clarabelle
Bonzo	Clarabelle
Mister Sniffles	Mister Sniffles



Suponha que você tenha tabelas idênticas, `palhaco_info1` e `palhaco_info2`. Escreva conexão única para obter uma tabela de resultado contendo o nome de cada palhaço e o nome do chefe de cada palhaço.

`palhaco_info1`

id	nome	id_chefe
1	Elsie	3
2	Pickles	5
3	Snuggles	10
4	Mr. Hobo	3
5	Clarabelle	10
6	Scooter	3
7	Zippo	3
8	Babe	5
9	Bonzo	5
10	Mister Sniffles	10

`palhaco_info2`

id	nome	id_chefe
1	Elsie	3
2	Pickles	5
3	Snuggles	10
4	Mr. Hobo	3
5	Clarabelle	10
6	Scooter	3
7	Zippo	3
8	Babe	5
9	Bonzo	5
10	Mister Sniffles	10



## Aponte seu lápis

### Solução

Suponha que você tenha tabelas idênticas, palhaco\_info1 e palhaco\_info2. Escreva conexão única para obter uma tabela de resultado contendo o nome de cada palhaço e o nome do chefe de cada palhaço.

palhaco\_info1

id	nome	id_chefe
1	Elsie	3
2	Pickles	5
3	Snuggles	10
4	Mr. Hobo	3
5	Clarabelle	10
6	Scooter	3
7	Zippo	3
8	Babe	5
9	Bonzo	5
10	Mister Sniffles	10

palhaco\_info2

id	nome	id_chefe
1	Elsie	3
2	Pickles	5
3	Snuggles	10
4	Mr. Hobo	3
5	Clarabelle	10
6	Scooter	3
7	Zippo	3
8	Babe	5
9	Bonzo	5
10	Mister Sniffles	10

```
SELECT p1.nome, p2.nome AS chefe
FROM palhaco_info1 p1 INNER JOIN
palhaco_info2 p2 ON p1.id_chefe = p2.id;
```

É aqui que combinamos o id\_chefe  
da palhaco\_info1 com o id de  
palhaco\_info2.

Assim não iremos nos confundir  
por duas colunas chamadas 'nome',  
nós colocaremos o pseudônimo na  
segunda como 'chefe'.

## Nós precisamos de uma auto relacionamento (SELF-JOIN)

No exercício “Aponte seu lápis” foi justamente o que você fez, recebeu a mesma tabela duas vezes. Mas em um banco de dados normalizado, você nunca teria duas cópias da mesma tabela. Ao invés disso, podemos **utilizar um auto relacionamento para a simulação de termos duas tabelas**.

Leve em consideração esta consulta que é quase idêntica a solução do “Aponte seu lápis”, mas tem uma diferença óbvia.

```
SELECT p1.nome, p2.nome AS chefe
FROM palhaco_info1 p1 ←
INNER JOIN palhaco_info2 p2 ←
ON p1.id_chefe = p2.id;
```

Estamos utilizando a tabela palhaco\_info duas vezes. Ela obteve o pseudônimo de c1 (onde pegaremos o id\_chefe) e c2 (onde obteremos o nome do chefe).

palhaco\_info

id	nome	id_chefe
1	Elsie	3
2	Pickles	5
3	Snuggles	10
4	Mr. Hobo	3
5	Clarabelle	10
6	Scooter	3
7	Zippo	3
8	Babe	5
9	Bonzo	5
10	Mister Sniffles	10

através de duas tabelas idênticas, nós utilizamos `palhaco` duas vezes, primeiro com o pseudônimo `p1` e depois com `p2`. Então estamos fazendo uma conexão interna para conectar `id_chefe` (de `p1`) com o nome do chefe (de `p2`).

nome	chefe
Elsie	Snuggles
Pickles	Clarabelle
Snuggles	Mister Sniffles
Mr. Hobo	Snuggles
Clarabelle	Mister Sniffles
Scooter	Snuggles
Zippo	Snuggles
Babe	Clarabelle
Bonzo	Clarabelle
Mister Sniffles	Mister Sniffles

Esta coluna vem da CONEXÃO INTERNA de `id_chefe` na primeira instância da tabela `palhaco_info` (`p1`) e o nome daquele chefe da segunda instância da tabela `palhaco_info` (`p2`).

A autocoluna permite que você consulte uma só tabela embora existam duas tabelas exatamente com a mesma informação em ambas.

## Outra forma de obter informações multi-tabelas



Estou tentando obter uma grande lista de todos os cargos que utilizei na `gregs_list`, mas não consigo descobrir como listar todos os cargos naquelas três tabelas de uma vez.

Estas são as três tabelas que Greg está falando.

Emprego que a pessoa possui atualmente.

emprego_atual
<code>id_contato</code> ↗
<code>cargo</code>
<code>salario</code>
<code>data_inicio</code>

Emprego que a pessoa deseja

emprego_desejado
<code>id_contato</code> ↗
<code>cargo</code>
<code>salario_minimo</code>
<code>salario_maximo</code>
<code>disponibilidade</code>
<code>anos_exp</code>

Empregos disponíveis

lista_empregos
<code>id_emploi</code> ↗
<code>cargo</code>
<code>salario</code>
<code>cep</code>
<code>descricao</code>

Até agora ele criou três comandos SELECT separados:

```
SELECT cargo FROM emprego_atual;
SELECT cargo FROM emprego_desejado;
SELECT cargo FROM lista_empregos;
```

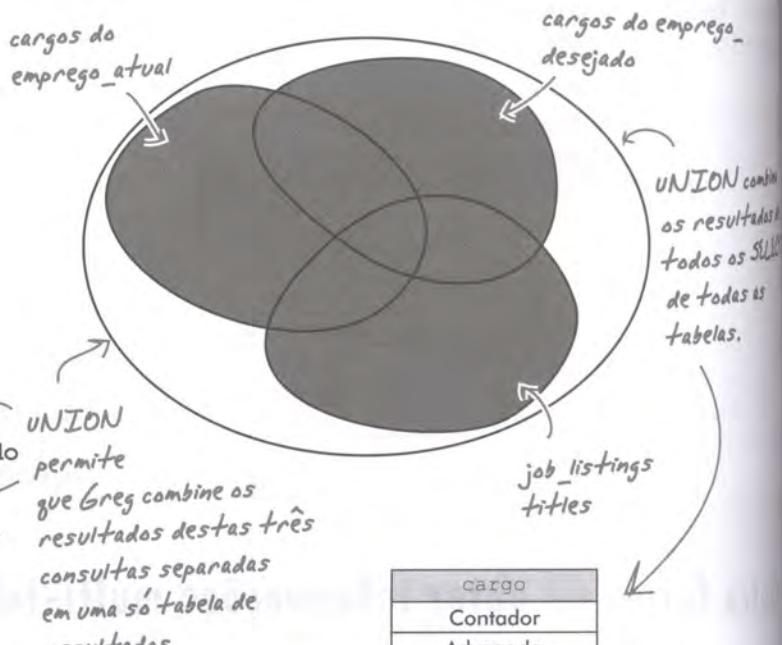
Elas funcionam, mas ele quer combinar os resultados em uma só consulta e obter uma lista de cada cargo listado naquelas três tabelas.

# Você pode utilizar uma UNION (UNIÃO)

Há outro jeito de obter resultados combinados de duas ou mais tabelas chamado de UNION.

Uma UNION combina de duas ou mais consultas em uma tabela baseada no que você especifica na lista de colunas do SELECT. Pense nos resultados de uma UNION como se fossem os resultados de cada SELECT que se “sobrepusem”.

```
SELECT cargo FROM emprego_atual
UNION
SELECT cargo FROM emprego_desejado
UNION
SELECT cargo FROM lista_empregos;
```



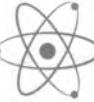
Greg percebeu que não há nenhuma duplicidade nos resultados, mas os cargos não estão em ordem, então ele tenta utilizar a consulta novamente com um ORDER BY em cada comando SELECT.

Estes são apenas algumas das centenas de linhas que ele obteve como resultado nos resultados combinados das três tabelas.

cargo
Contador
Advogado
Programador
Web Designer
Pastor de Gatos
Chef
Psicólogo

```
SELECT cargo FROM emprego_atual ORDER BY cargo
UNION
SELECT cargo FROM emprego_desejado ORDER BY cargo
UNION
SELECT cargo FROM lista_empregos ORDER BY cargo;
```

Greg adicionou um ORDER BY em cada comando para que os cargos na tabela resultados sejam listados alfabeticamente.



**PODER DO CÉREBRO**

O que você acha que aconteceu quando Greg executou esta nova consulta?

## UNION é limitado

Consulta de Greg não funcionou! Greg obteve um erro porque seu software não sabia como interpretar os múltiplos ORDER BY.

UNION só pode ter um ORDER BY **ao final do comando**. Isto porque UNION **concatena e agrupa os resultados dos múltiplos comandos SELECT**.

Além disso, há mais algumas coisas sobre uniões que você deveria saber.

### Regras de UNION para SQL

O número de colunas em cada comando SELECT deve **combinar**. Você não pode selecionar duas colunas do primeiro comando e uma do próximo.

Você deve ter também as mesmas expressões e agregar funções em cada comando SELECT.

Você pode colocar o comando SELECT em qualquer ordem; isto não alterará os resultados.

### Regras de UNION para SQL

Por padrão, SQL suprime valores duplicados dos resultados da UNION.

Os tipos de dados nas colunas precisam ser os mesmos ou serem conversíveis entre si.

Se por alguma razão você quer ver os resultados duplicados, você pode utilizar o operador UNION ALL. Ele retorna cada resultado, não só os diferentes.

## Regras do UNION em ação

O número de colunas nos comandos SELECT que você está combinando com UNION devem combinar. Você não pode usar o SELECT com duas colunas da primeira tabela e uma só coluna da próxima tabela.

Você deve utilizar o mesmo número de colunas em cada SELECT.

```
SELECT cargo FROM emprego_atual
UNION
SELECT cargo FROM emprego_desejado
UNION
SELECT cargo FROM lista_empregos
ORDER BY cargo;
```

Se quiser ordenar os resultados, utilize um ORDER BY depois do último SELECT que você está combinando. Isto ordena toda a lista de resultados.

cargo
Confeiteiro
Pastor de Gatos
Caubói de Gatos
Palhaço
Treinador de Cães
Cabeleireiro
Joalheiro
Advogado
Mecânico
Neurocirurgião

Aqui está um exemplo dos resultados que podemos obter.

Neste exemplo, todas as três colunas têm o mesmo tipo de dados: VARCHAR. Como um resultado, a coluna retornada pela consulta é também uma VARCHAR.



### PODER DO CÉREBRO

O que você acha que aconteceria se as colunas unidas tivessem tipos de dados diferentes?

## UNION ALL

UNION ALL funciona exatamente como o UNION, exceto que eles retornam todos os valores das colunas, ao invés de apenas uma instância de cada valor que é duplicado.

Desta vez queremos ver todos os valores armazenados

```
SELECT cargo FROM emprego_atual
UNION ALL
SELECT cargo FROM emprego_desejado
UNION ALL
SELECT cargo FROM lista_empregos
ORDER BY cargo;
```

cargo
Confeiteiro
Confeiteiro
Pastor de Gatos
Caubói de Gatos
Palhaço
Palhaço
Palhaço
Treinador de Cães
Treinador de Cães
Cabeleireiro
Joalheiro
Advogado
Advogado
Advogado
Mecânico
Neurocirurgião

Desta vez obtemos os mesmos cargos listados mais de uma vez.

Até agora nossas UNIONs têm usado colunas do mesmo tipo de dados. Mas você poderá criar UNION de colunas de diferentes tipos de dados.

Quando dizemos que os tipos de dados devem ser conversíveis em para o outro, queremos dizer que os tipos de dados retornados serão convertidos em um tipo de dados compatível, se possível, e se não for, a consulta irá falhar.

Suponha que você usou UNION em dados do tipo INTEGER e VARCHAR. Já que VARCHAR não pode se tornar uma integer, as linhas do resultado serão convertidas de INTEGER para VARCHAR.

## Crie uma tabela a partir da sua UNION

Para conseguirmos saber qual é o tipo de dados retornado pela nossa UNION, a não ser que nós capturemos isso de alguma forma. Podemos utilizar uma CREATE TABLE AS para apanhar nossos resultados da UNION e observá-lo mais de perto.

O comando CREATE TABLE AS pega o resultado de uma consulta SELECT e cria uma tabela a partir destes resultados. No exemplo abaixo, estamos colocando nosso resultado UNION em uma nova tabela chamada minha\_union.

```
O nome da nossa nova tabela. →
CREATE TABLE minha_union AS
SELECT cargo FROM emprego_atual UNION
SELECT cargo FROM emprego_desejado
UNION SELECT cargo FROM lista_empregos;
```

← Esta é a UNION que já vimos. Você pode criar uma tabela a partir de qualquer comando SELECT.

 Aponte seu lápis —

Crie uma UNION a partir do seguinte: id\_contato de emprego\_atual e salário de lista\_empregos

.....

Dê um palpite de como serão os tipos de dados dos resultados, depois escreva um comando CREATE TABLE AS com sua UNION.

.....

Faça um DESC de sua tabela e veja se estava certo sobre os tipos de dados.

.....

→ Respostas na página 366.

## INTERSECT e EXCEPT

INTERSECT e EXCEPT são usados quase da mesma forma que UNION - para encontrar partes de consultas sobrepostas.

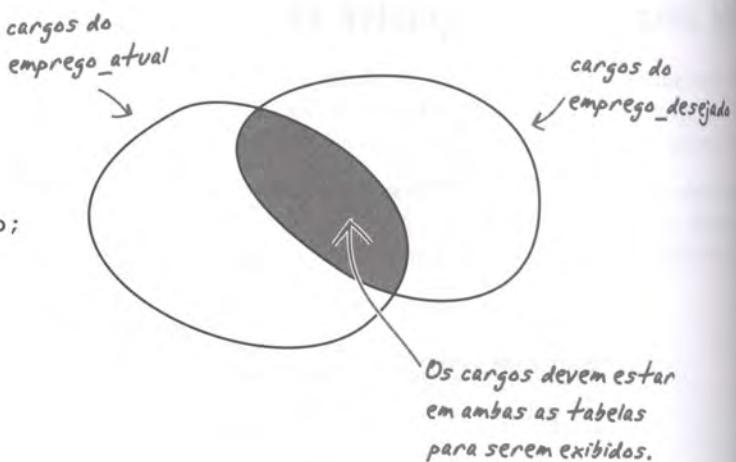
INTERSECT retorna apenas os valores que estão na primeira consulta e na segunda consulta.



**Veja Isto!**

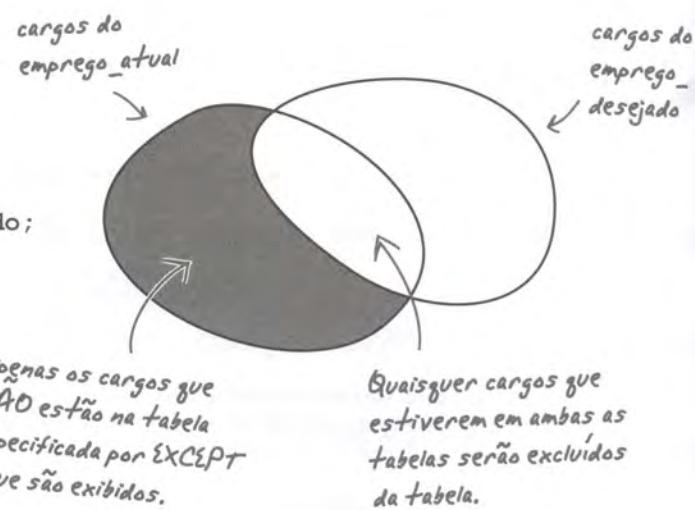
Estas duas operações não existem no MySQL.

```
SELECT cargo FROM emprego_atual
INTERSECT
SELECT cargo FROM emprego_desejado;
```



EXCEPT retorna aquelas colunas que estão na primeira consulta, mas **não** na segunda coluna.

```
SELECT cargo FROM emprego_atual
EXCEPT
SELECT cargo FROM emprego_desejado;
```



Nós já tivemos o suficiente de conexões, hora de ir para



Espere um pouco. Você não pode fazer todo este suspense. Você disse que conexões e subconsultas fazem a mesma coisa. Você precisa provar isto.

(Derrrr, certo, o que nós queríamos dizer era...)

## Subconsultas e conexões comparadas

Praticamente, qualquer coisa que você possa fazer com uma subconsulta, poderá fazer com uma conexão. Vamos retroceder algumas páginas até o começo do Capítulo 9.

**Subconsultas**

Para conseguir fazer o que aquelas duas consultas fazem, em uma só consulta, precisamos adicionar uma subconsulta na nossa consulta.

Nós chamaremos a *segunda* consulta que usamos para obter as combinações da tabela *profissão* a consulta EXTERNA, porque ela cobrirá dentro de si a consulta INTERNA. Vamos ver como isso funciona.

**Consulta Externa**

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
WHERE
    ea.cargo IN ('Cozinheira', 'Cabeleireira', 'Garçom', 'Web designer', 'Programador em Web');
```

Esta parte é a consulta externa.

Esta parte pode ser removida e recalculada como parte de nossa primeira consulta, a que se tornará a consulta interna.

Toda essa profisões acima, em parênteses, vêm da primeira consulta que fizemos, aquela que selecionou todos os cargos da tabela *emprego\_atual*. Então - essa é a parte inteligente, então preste atenção - nós podemos trocar de lugar aquela parte da consulta externa com parte da primeira consulta. Isto ainda vai produzir todos os resultados entre os parenteses acima, mas esta consulta agora fica entendida como a subconsulta:

**Consulta interna**

```
SELECT cargo FROM lista_empregos
GROUP BY cargo ORDER BY cargo;
```

Esta parte de primeira consulta se tornará a consulta interna ou subconsulta.

**Nós combinamos as duas em uma consulta com uma subconsulta**

Tudo o que fizemos foi combinar as duas consultas em uma só. A primeira consulta é conhecida como consulta externa. A outra dentro dela é conhecida como consulta interna.

**Consulta Externa** + **Consulta interna** = **Consulta com subconsulta**

*As duas consultas combinadas em uma só é uma consulta contendo uma subconsulta.*

**SELECT mc.primeiro\_nome, mc.sobrenome, mc.telefone, ea.cargo  
FROM emprego\_atual AS ea NATURAL JOIN meus\_contatos AS mc  
WHERE ea.cargo IN (SELECT cargo FROM lista\_empregos);**

*Não precisamos digitar novamente todas as profissões de nossa primeira consulta porque a consulta interna já temos conta disso por nós!*

*Mesmos resultados que anteriormente só que utilizando uma só consulta!*

mc.primeiro_nome	mc.sobrenome	mc.telefone	ea.cargo
Joe	Lonnigan	(555) 555-3214	Cozinheira
Wendy	Hillerman	(555) 555-8976	Garçom
Sean	Miller	(555) 555-4443	Web designer
Jared	Callaway	(555) 555-5674	Programador em Web
Juan	Garza	(555) 555-0098	Programador em Web

você está aqui ▶ 313      314 Capítulo 9

## Transformando uma subconsulta em uma conexão

Volta ao Capítulo 9, esta foi a primeira subconsulta que criamos:

*Consulta externa.*

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
WHERE ea.cargo IN (SELECT cargo FROM lista_empregos);
```

*Consulta interna.*

Estes são os resultados que obtemos quando nós executamos nossa consulta:

mc.primeiro_nome	mc.sobrenome	mc.telefone	ea.cargo
Joe	Lonnigan	(555) 555-3214	Cozinheira
Wendy	Hillerman	(555) 555-8976	Garçom
Sean	Miller	(555) 555-4443	Web designer
Jared	Callaway	(555) 555-5674	Programador em Web
Juan	Garza	(555) 555-0098	Programador em Web

você está aqui ▶ 359



## Aponte seu lápis

Aqui está a cláusula WHERE com a subconsulta escrita como uma CONEXÃO INTERNA:

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
INNER JOIN lista_empregos le
ON ea.cargo = le.cargo;
```

Você pode substituir a cláusula WHERE, que contém a subconsulta, por uma INNER JOIN (CONEXÃO INTERNA).

Explique o porquê de esta parte da CONEXÃO INTERNA da consulta dará os mesmos resultados que uma subconsulta.

Quais destas consultas você achou mais fácil de entender?

Respostas na página 364.

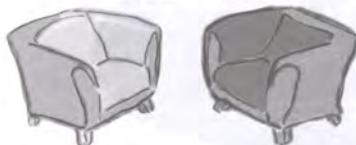


Se eu já tenho tudo escrito utilizando subconsultas, devo voltar atrás e reescrevê-las como conexões?

**Não. Se você já aquelas subconsultas fazendo o que elas devem fazer, você não precisa reescrevê-las.**

Mas definitivamente há razões para escolher uma ao invés da outra algumas vezes...

## Bate-papo



### Conversa de hoje:

Conexão versus Subconsulta, qual é a melhor

## Conexão

Eu sou claramente a melhor opção na maioria dos casos. Eu sou mais fácil de se entender, e geralmente executo minha tarefa muito mais rapidamente que a velha Subconsulta.

Eu estava indo muito bem sem você. Eu sou mais fácil de ser compreendida que você.

Diga você. E quanto aquelas bobeiras de correlacionado ou não-correlacionado?

## Subconsulta

Com licença? Quem você está chamando de “velha”? Eu nem existia até a criação de alguns Sistemas SQL. Eu fui adicionada porque muitos programadores queriam me utilizar.

Quem você está tentando enganar, com suas besteiras de EXTERNA e INTERNA? Isso é tudo muito confuso...

→ Confina na próxima página

**Bate-papo****Conexão**

Nem sempre, Senhora Subconsulta correlacionada. Mas tudo bem, vamos deixar isso pra lá, por ora. Eu sou a melhor opção quando você precisa de colunas de múltiplas tabelas em seus resultados. De fato, eu sou a única opção quando você precisa disto.

Isto pode ser verdade, mas pelo menos não é tão difícil descobrir o que estou fazendo. Por que você até pode usar pseudônimos para evitar digitar os nomes das tabelas várias e várias vezes.

Blá blá blá. Bom demais para os pseudônimos, não é? E você se acha tão mais simples do que eu, mas e quanto aquelas subconsultas correlacionadas? Elas são muito mais enroladas do que qualquer coisa que eu possa fazer.

Exibida.

**Conversa de hoje:**

Conexão versus Subconsulta, qual é a melhor

**Subconsulta**

Ok, nós dois temos os nossos jargões; isto é verdade. Mas comigo, você pode simplesmente descobrir a parte interna e a externa separadamente.

Qual é porque você não é assim bom com valores agregados. Você não pode usar agregados na cláusula WHERE sem uma subquery. Esse faz um bocado para não retornar colunas múltiplas. Você é tão complicado.

Yeah, sobre aqueles pseudônimos, eu penso que fazem coisas mais ainda mais difíceis de seguir. E para o registro, eu posso usá-los bastante, você sabe. Mas quando eu os uso, é muito mais direto. Metade do tempo onde eu não me incomodo mesmo com pseudônimos.

Errr... Verdade. Mas eu sei uma coisa que me faz muito diferente de você. Eu posso ser usado com UPDATE, INSERT, e DELETE.



Pegue estas consultas com subconsultas do Capítulo 9 e veja se consegue reescrevê-las sem as subconsultas, ou se ficar melhor deixar as subconsultas na sua consulta. Conexões são permitidas.

Liste os cargos que ganham salário igual ao maior salário da tabela lista\_emploies.

```
SELECT cargo FROM lista_emploies WHERE salario = (SELECT
MAX(salario) FROM lista_emploies);
```

**Melhor utilizar subconsultas?**

Liste o primeiro nome e o sobrenome das pessoas com o salário maior que a média.

```
SELECT mc.primeiro_nome, mc.sobrenome FROM meus_contatos
mc NATURAL JOIN emprego_atual ea WHERE ea.salario >
(SELECT AVG(salario) FROM emprego_atual);
```

**Melhor utilizar subconsultas?**



Pegue estas consultas com subconsultas do Capítulo 9 e veja se consegue reescrevê-las sem as subconsultas, ou se ficar melhor deixar as subconsultas na sua consulta. Conexões são permitidas.

### Exercícios

Liste os cargos que ganham salário igual ao maior salário da tabela lista\_emploies.

```
SELECT cargo FROM lista_emploies WHERE salario = (SELECT
MAX(salario) FROM lista_emploies);
```

*SELECT cargo FROM lista\_emploies*

*By salario DESC LIMIT 1;*

*Isto faz com a consulta retornar apenas um valor, a linha com o maior salário.*

**Melhor utilizar subconsultas?** *Não.*

Liste o primeiro nome e o sobrenome das pessoas com o salário maior que a média.

```
SELECT mc.primeiro_nome, mc.sobrenome FROM meus_contatos
mc NATURAL JOIN emprego_atual ea WHERE ea.salario >
(SELECT AVG(salario) FROM emprego_atual);
```

*Ob, oh, não podemos utilizar LIMIT e ORDER BY para obter resultados que são as médias, da mesma forma que fizemos na consulta acima.*

**Melhor utilizar subconsultas?**

*Sim.*

*Na solução anterior, fomos capazes de utilizar LIMIT para obter os maiores salários a partir de uma lista ordenada da coluna salario. Os nossos salários maiores não podem ser ordenados, então não há como usar LIMIT com eles.*

## Auto relacionamento como uma subconsulta

Quanto você estava vendo como transformar sua subconsulta em conexão, vamos ver como transformar uma autoconexão em uma consulta.

Quais são da `id_palhaco_chefe` que adicionamos na nossa tabela `palhaco_info`? Aqui está o auto relacionamento que utilizamos onde temos uma instância da tabela `palhaco_info` 1 ou `p1`, e a segunda

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

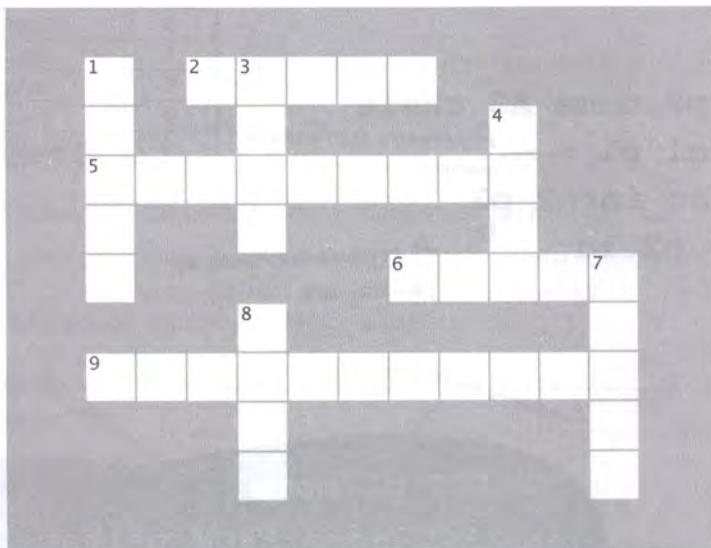
372

373



## Cruzadas de Conexões e Unions

Este foi um capítulo turbinado, com muita coisa para se aprender. Ajude a assimilar tudo fazendo estas palavras-cruzadas. Todas as respostas estão neste capítulo.



### Horizontais

2. Esta combina o resultado de duas ou mais consultas em uma tabela, baseado no que você especifica na lista de colunas de seu comando SELECT.
5. Por padrão, SQL suprime valores \_\_\_\_\_ dos resultados de sua UNION.
6. Uma conexão \_\_\_\_\_ dá uma linha, independentemente de haver ou não uma combinação na outra tabela.
9. Uma chave externa auto-\_\_\_\_\_ é a chave primária de uma tabela utilizada naquela mesma tabela para outro propósito.

### Verticais

1. Com uma conexão interna, você está comparando linhas de duas tabelas, mas a \_\_\_\_\_ destas duas tabelas não importa.
3. Este é o resultado de uma consulta externa esquerda que significa que a tabela direita não tem nenhum valor correspondente na tabela esquerda.
4. Uma Consulta Externa \_\_\_\_\_ pega todas as linhas da tabela esquerda e as combina com as linhas da tabela direita.
7. Uma consulta externa \_\_\_\_\_ avalia a tabela direita em relação à tabela esquerda.
8. Nós podemos usar uma \_\_\_\_\_ - conexão para simular ter duas tabelas.

→ Continua na próxima página



## Sua caixa de ferramentas SQL

Você está realmente indo agora. Você já viu as conexões externas, auto relacionamento, unions e já até sabe como converter uma conexão em uma subconsulta, e vice-versa. Para uma lista completa de dicas neste livro, veja o Apêndice iii.

### CHAVE ESTRANGEIRA AUTOREFERENCIADA

Esta é uma chave externa na mesma tabela onde se encontra a chave primária, utilizada para outro propósito.

### UNION e UNION ALL

O UNION combina os resultados de duas ou mais consultas em uma tabela, baseando-se no que você especifica na lista de colunas do SELECT.

UNION esconde os valores duplicados.  
UNION ALL inclui os valores duplicados.

### CONSULTA EXTERNA ESQUERDA (LEFT OUTER JOIN)

Uma CONSULTA EXTERNA ESQUERDA pega todas as linhas de uma tabela esquerda e as combina com as linhas da tabela DIREITA.

### AUTO RELACIONAMENTO (SELF-JOIN)

O AUTO RELACIONAMENTO permite que você consulte uma só tabela ainda que existam duas tabelas contendo a mesma informação nelas.

### CREATE TABLE AS

Utilize este comando para criar uma tabela a partir do resultado de qualquer comando SELECT.

### CONEXÃO EXTERNA DIREITA (RIGHT OUTER JOIN)

Uma CONEXÃO EXTERNA DIREITA pega todas as linhas de uma tabela direita e combina com todas as linhas da tabela ESQUERDA.

### INTERSECT

utilize esta palavra-chave apenas para valores que estão na primeira consulta e também na segunda consulta.

### EXCEPT

utilize esta palavra-chave para retornar apenas os valores que estão na primeira consulta, MAS NÃO estão na segunda consulta.



**Aponte seu lápis**  
**Solução**  
Da página 357.

Crie uma UNION a partir do seguinte: id\_contato de emprego\_atual e salário de lista\_employos

```
SELECT id_contato FROM emprego_atual
UNION SELECT salario FROM lista_employos;
```

Dê um palpite de como serão os tipos de dados dos resultados, depois escreva um comando CREATE TABLE AS com sua UNION.

```
CREATE TABLE minha_tabela SELECT
Id_contato FROM emprego_atual UNION
SELECT salario FROM lista_employos;
```

Faça um DESC de sua tabela e veja se estava certo sobre os tipos de dados.

DEC(12,2)



**Aponte seu lápis**

**Solução** Aqui está a cláusula WHERE com a subconsulta escrita como uma CONEXÃO INTERNA:  
Da página 360.

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.telefone, ea.cargo
FROM emprego_atual AS ea NATURAL JOIN meus_contatos AS mc
INNER JOIN lista_employos le
ON ea.cargo = le.cargo;
```

Você pode substituir a cláusula WHERE, que contém a subconsulta, por uma INNER JOIN (CONEXÃO INTERNA).

Explique o porquê de esta parte da CONEXÃO INTERNA da consulta dará os mesmos resultados que uma subconsulta.

A Conexão interna exibe resultados apenas quando ea.cargo = le.cargo, o que é equivalente à cláusula WHERE com a subconsulta:  
WHERE ea.cargo IN (SELECT cargo FROM lista\_employos);

Quais destas consultas você achou mais fácil de entender?

Não há uma resposta correta aqui! Mas sua resposta mostra que você está começando a considerar o que irá utilizar no futuro com seus próprios dados.



## Solução da Cruzada de Conexões e Unions

Solução da página 364.

<sup>1</sup> O	<sup>2</sup> U	<sup>3</sup> N	I	O	N		
R		V			L		
D	U	P	L	I	C	A	T
E			L			F	
R				O	U	T	E
				R			
					S		I
						H	
						T	



## 11 Constraints, views e transactions

# \*Cozinhar demais pode estragar o banco de dados

Viu? É aqui que você errou.  
Para "quantidade" você  
colocou "um montão".

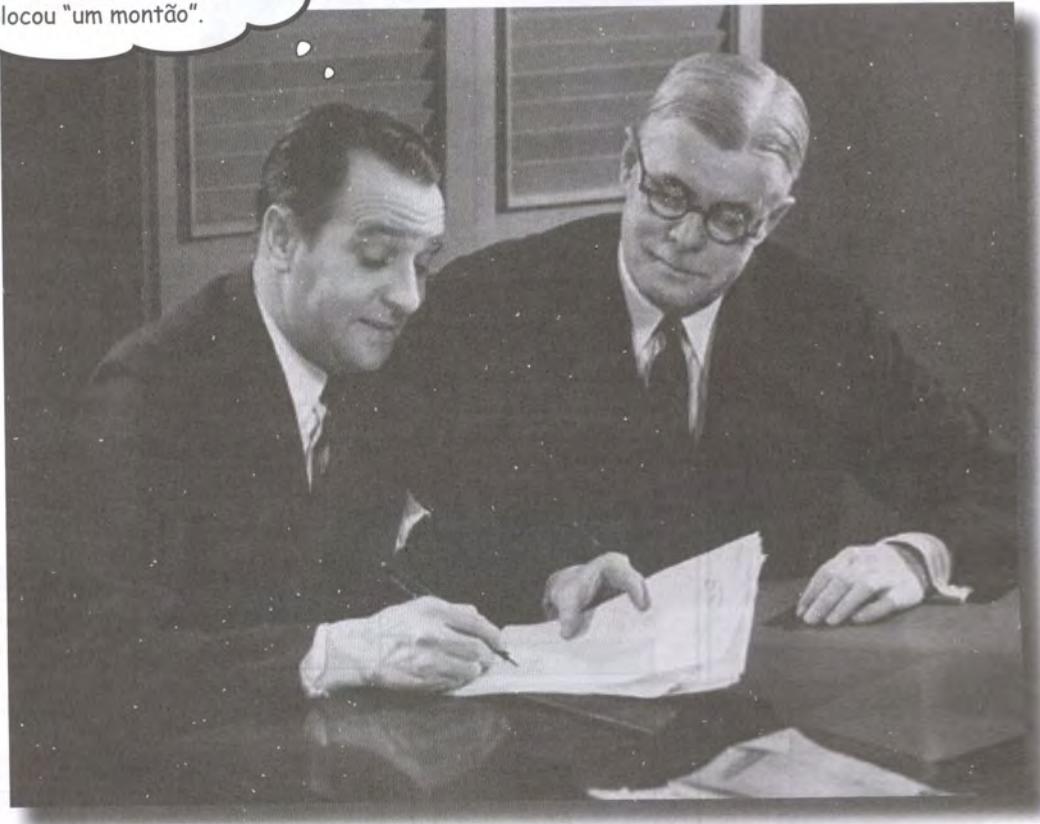


**Seu banco de dados cresceu, e outras pessoas precisam utilizá-lo.** O problema é que alguns deles não serão tão hábeis como você é com SQL. Você precisa de algumas maneiras de **impedir os de acessar dados errados**, técnicas para permiti-los **visualizar somente parte dos dados**, e maneiras de **pisarem um nos outros ao tentarem acessar os mesmos dados ao mesmo tempo**. Neste capítulo, começamos a proteger nossos dados do erro de outros. Bem-vindo aos Bancos de dados defensivos, Parte 1.

## 11 Constraints, views e transactions

### \* Cozinhar demais pode estragar o banco de dados

Viu? É aqui que você errou.  
Para "quantidade" você  
colocou "um montão".

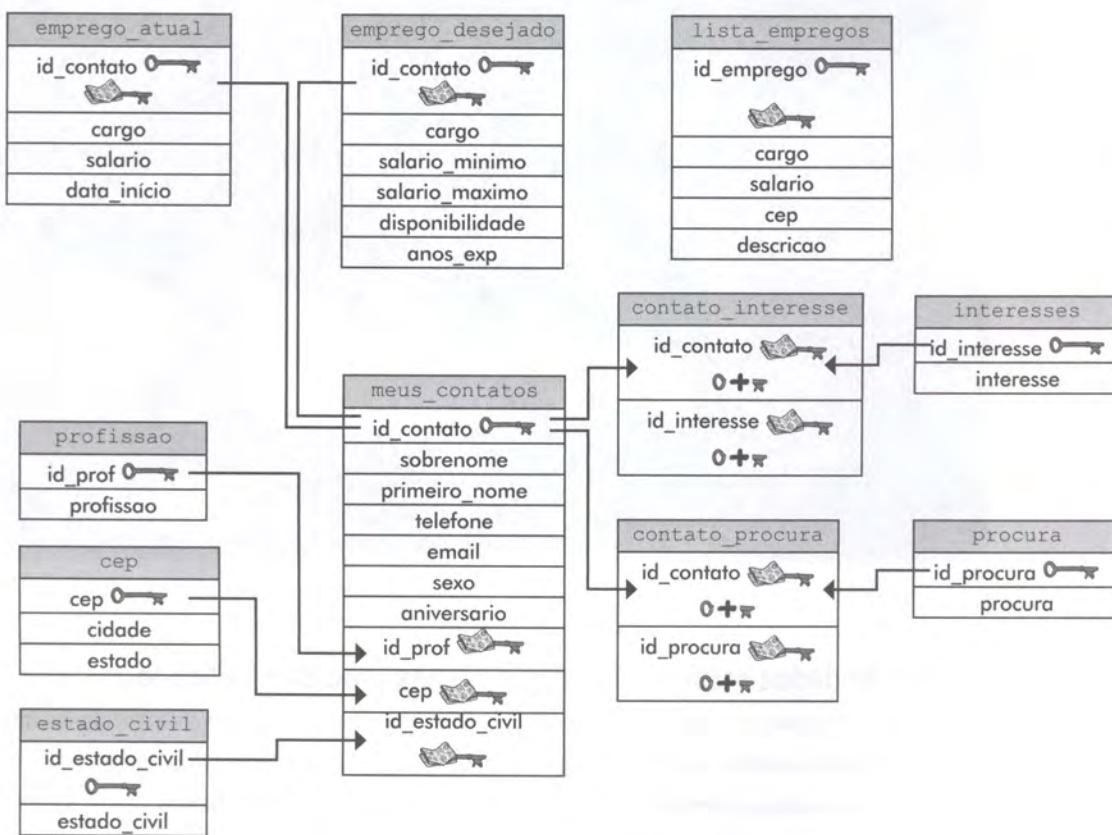


**Seu banco de dados cresceu, e outras pessoas precisam utilizá-lo.** O problema é que alguns deles não serão tão hábeis como você é com SQL. Você precisa de algumas maneiras de **impedir** os deles de acessar dados errados, técnicas para permiti-los visualizar somente parte dos dados, e maneiras de pisarem um nos outros ao tentarem acessar os mesmos dados ao mesmo tempo. Neste capítulo, começamos a proteger nossos dados do erro de outros. Bem-vindo aos Bancos de dados defensivos, Parte 1.

## Greg contratou ajuda

Greg contratou duas pessoas para ajudá-lo a gerenciar seu negócio em ascensão. Jim ficará responsável por inserir novos clientes no Banco de Dados, enquanto Frank é responsável por combinar clientes com prósperos empregos.

Greg gastou algum tempo explicando seu banco de dados para eles e explicando o que cada tabela faz.



## Primeiro dia de Jim: Inserindo um novo cliente

Sentado em seu novo cubículo e recebe uma mensagem de Greg:

Chat do Time: Aqui estão os dados a serem inseridos

Hey Jim, você pode adicionar alguém no banco de dados para mim?

Sim, claro.

Esta é apenas parte da informação. Vou passar o resto depois:

Pat Murphy, 555-1239, patmurphy@someemail.com, o cep é 10087

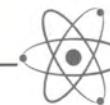
15/04/1978 é o aniversário

Para profissão, utilize professor, e estado-civil casado (você terá que fazer alguns SELECTs para colocar os valores corretos aqui. Olhe minhas anotações para verificar a sintaxe).

Parece fácil. Deixa comigo. :)

Obrigado!

De nada



### PODER DO CÉREBRO

Você pode escrever as consultas para inserirem as novas pessoas no banco de dados?

## Jim evita NULL

À medida que ele insere os dados, Jim percebe que ele não sabe se Pat é homem ou mulher. Greg não está por perto, então ele toma uma decisão de comando. Ele decide inserir 'X' para sexo.

Aqui estão suas consultas:

### Ele obtém o id\_prof da tabela profissão.

```
SELECT id_prof FROM profissao WHERE profissao = 'professor';
```

id_prof
19

Aqui está o id que corresponde a professor, então ele pode utilizar esta informação em sua consulta a tabela meus\_contatos.



### Ele obtém o id\_estado\_civil da tabela estado\_civil

```
SELECT id_estado_civil FROM profissao WHERE estado_civil = 'solteiro';
```

id_estado_civil
4

E aqui está o id\_estado\_civil que corresponde a casado.

meus_contatos
id_contato
sobrenome
primeiro_nome
telefone
email
sexo
aniversario
id_prof
cep
id_estado_civil

### Ele insere estes valores e coloca 'X' para sexo

```
INSERT INTO meus_contatos VALUES('', 'Murphy', 'pat',
'5551239', 'patmurphy@someemail.com', 'X', '15-04-1978',
19, '10087', 3);
```

↑              ↑  
Estes são os IDs que ele encontrou com as duas consultas acima. Ele poderia ter feito com subconsultas.

Quando temos uma coluna AUTO\_INCREMENT, nós não precisamos inserir um valor nela. Estas duas aspas dizem a tabela para inserir um valor para a coluna chave primária.

↑  
Esta é a informação que Jim decidiu inserir para sexo ao invés de tentar adivinhar ou ainda inserir informação do tipo NULL.

## Antecipando três meses

Greg está tentando descobrir alguns dados demográficos. Ele quer saber quantas pessoas em meus\_contatos são do sexo masculino, quantas do sexo feminino e quantos registros estão inseridos no total. Ele faz três consultas: primeiro ele obtém uma contagem de todos os registros com sexo masculino e feminino, depois ele obtém a contagem do total de registros.

```
SELECT COUNT(*) AS Feminino FROM meus_contatos WHERE sexo = 'F';
```

Feminino
5975

Greg descobriu que ele possui 5795 linhas com o sexo 'F' na tabela meus\_contatos.

SELECT COUNT(\*) AS Masculino FROM meus\_contatos WHERE sexo = 'M';

Masculino
6982

← 6982 valores como sexo 'M'.

SELECT COUNT(\*) AS Total FROM meus\_contatos;

Total
12970

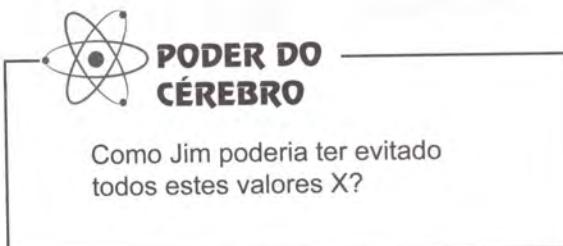
← Ele checa o número total de linhas em sua tabela com esta consulta.

parece que o número de consultas não batem. Ele possui 13 linhas que aparentemente parecem sob a consulta de masculino ou feminino. Ele tenta outra consulta.

SELECT sexo FROM meus\_contatos  
WHERE sexo <> 'M' AND sexo <> 'F';

sexo
X
X
X
X
X
X
X
X
X
X
X
X
X

← Quando ele olha os registros que faltavam, ele localiza o 'X' como valor para sexo.



## CHECK, por favor: Adicionando uma CHECK CONSTRAINT (restrição CHECK)

vimos uma série de constraints em colunas nos capítulos anteriores. constraint é uma restrição na qual você pode inserir em uma coluna. constraints são adicionadas quando criamos a tabela. Algumas das constraints que já viu incluem NOT NULL, PRIMARY KEY, FOREIGN KEY e UNIQUE. Um outro tipo de constraints para colunas, chamadas CHECK. Aqui está um exemplo de uma. Suponha que temos um cofrinho de moedas e queremos manter estoque das moedas depositadas nele. Ele só recebe moedas de Um, Cinco, Dez e cinco centavos. Podemos utilizar as letras U, C, D e V para representar tipo de moeda. A tabela abaixo utiliza a CHECK CONSTRAINT para restringir os valores que possam ser inseridos na coluna de moedas:

CREATE TABLE cofrinho

```
(  
    id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,  
    moeda CHAR(1) CHECK (coin IN ('U', 'C', 'D', 'V'))  
)
```

← Esta parte checa para verificar se a coluna moeda é alguma destas.

Uma CHECK CONSTRAINT restringe quais valores você pode inserir na coluna. Ela utiliza as mesmas condicionais que uma cláusula WHERE.

**Se o valor que está tentando inserir falhar a esta condição, você vai obter uma mensagem de erro.**



### Veja Isto!

**CHECK doesn't enforce data integrity in MySQL**

Você pode criar com **CHECK CONSTRAINTS** no MySQL, mas ela não fará nada por você. MySQL as ignora.

## CHECKando o sexo

Se Greg pudesse voltar no tempo, ele poderia ter criado a tabela `meus_contatos` com uma CHECK CONSTRAINT na coluna `sexo`. Ao invés disso, ele pode consertar o problema com uma `ALTER TABLE`.

```
ALTER TABLE meus_contatos
ADD CONSTRAINT CHECK sexo IN ('M', 'F');
```

No dia seguinte Jim se vê impossibilitado de inserir 'X' para sexo. Quando ele pergunta ao Greg o porquê disto, Greg explica sobre a nova constraint e diz ao Jim que já que não pode voltar ao tempo, ele fará com que Jim entre em contato com todos os registros inseridos como 'X' para descobrir qual sexo de cada um deles.



**Aponte seu lápis**

Escreva os valores que acha que são permitidos para cada uma destas colunas.

```
CREATE TABLE tabela_misteriosa
(
    coluna1 INT(4) CHECK (coluna1 > 200),
    coluna2 CHAR(1) CHECK (coluna2 NOT IN ('x', 'y', 'z')),
    coluna3 VARCHAR(3) CHECK ('A' = SUBSTRING(coluna_3, 1, 1)),
    coluna4 VARCHAR(3) CHECK ('A' = SUBSTRING(coluna_4, 1, 1)
        AND '9' = SUBSTRING(coluna_4, 2, 1))
)
```

Coluna 1: .....

Coluna 2: .....

Coluna 3: .....

Coluna 4: .....

*Por que continuo recebendo mensagens de erro?*



**Aponte seu lápis****Solução**

Escreva os valores que acha que são permitidos para cada uma destas colunas.

```
CREATE TABLE tabela_misteriosa
(
    coluna1 INT(4) CHECK (coluna1 > 200),
    coluna2 CHAR(1) CHECK (coluna2 NOT IN ('x', 'y', 'z')),
    coluna3 VARCHAR(3) CHECK ('A' = SUBSTRING(coluna_3, 1, 1)),
    coluna4 VARCHAR(3) CHECK ('A' = SUBSTRING(coluna_4, 1, 1)
    AND '9' = SUBSTRING(coluna_4, 2, 1))
)
```

*Você pode combinar as condições com AND ou OR.*

Coluna 1: Valores inseridos devem ser maiores que 200

Coluna 2: Quaisquer caracteres que não sejam x, y ou z podem ser inseridos.

Coluna 3: O primeiro caractere da linha de texto deve ser A

Coluna 4: O primeiro caractere da linha de texto deve ser A e o segundo deve ser 9.

## não existem Perguntas Idiotas

P: Então posso utilizar qualquer coisa na minha CHECK que usaria na minha cláusula WHERE?

R: Quase. Você pode utilizar todas as condicionais: AND, OR, IN, NOT, BETWEEN e outras. Ainda pode combiná-las, como viu no exemplo acima. No entanto, você não pode utilizar uma subconsulta.

P: Então, se eu não posso utilizá-las em MySQL, o que posso usar?

R: Não há uma resposta fácil para essa pergunta. Algumas pessoas utilizam os gatilhos, que nada mais são que consultas que serão executadas se uma certa condição é encontrada, mas elas não são tão fáceis quanto CHECK, e estão fora do foco deste livro.

P: O que acontece se você tenta utilizar o INSERT em valores que não satisfazem o CHECK?

R: Você vai obter um erro e nada será inserido.

P: Que bem faz isso?

R: Ela certifica que todos os dados inseridos na sua tabela façam sentido. Você não terá acabado ainda com os valores misteriosos.

## Frank trabalhou de Frank ficou Tedioso

Frank tem trabalhado combinando pessoas com os empregos. Ele está notando certos padrões. Ele tem várias vagas de emprego para web designers e não muitos candidatos. Ele tem muitos escritores técnicos procurando emprego, mas não há muitas posições abertas para eles.

Frank executa as mesmas consultas todos os dias para tentar encontrar as combinações para pessoas e vagas de emprego.

Tenho que criar as mesmas consultas, repetidamente, todos os dias. Isto é tedioso.



## Seja o Frank



Seu trabalho é interpretar Frank e escrever as consultas que Frank escreve todos os dias. Escreva uma consulta para encontrar todos os web designers da emprego\_desejado, bem como as informações para contato. Escreva outra consulta para encontrar vagas abertas para escritores técnicos.



## Solução do Seja o Frank

Seu trabalho é interpretar Frank e escrever as consultas que Frank escreve todos os dias. Escreva uma consulta para encontrar todos os web designers da emprego\_desejado, bem como as informações para contato. Escreva outra consulta para encontrar vagas abertas para escritores técnicos.

Estas não são consultas difíceis, mas ter que digitá-las diariamente, por diversas vezes, quase o obriga a cometer erros. Ele precisa de alguma forma para salvá-las e ver apenas o resultado uma vez por dia sem ter que digitá-las novamente.



Então ele pode simplesmente salvar suas consultas em um arquivo de texto e copiar e colá-las. Qual o grande problema?

### Arquivos podem ser sobreescritos ou modificados

O arquivo poderia ser accidentalmente modificado ou deletado. Há um jeito muito melhor de salvar estas consultas **dentro** do banco de dados. Podemos transformá-las em **views**.

```
SELECT mc.primeiro_nome, mc.sobrenome,  
mc.fone, mc.email  
FROM meus_contatos mc  
NATURAL JOIN emprego_desejado ed  
WHERE ed.cargo = 'Programador Web';
```

Greg geralmente deixa a maiúscula a descrição dos cargos no seu banco de dados.

```
SELECT cargo, salario, descricao, cep  
FROM lista_empregos  
WHERE cargo = 'Escritor Técnico';
```

## criando uma view

criar uma view é realmente simples. Nós adicionamos um comando CREATE VIEW em nossa consulta. Vamos criar duas views para as consultas de Frank:

```
CREATE VIEW web_designers AS
SELECT mc.primeiro_nome, mc.sobrenome, mc.fone, mc.email
FROM meus_contatos mc
```

```
NATURAL JOIN emprego_desejado ed
WHERE ed.cargo = 'Programador Web' ;
```

*Isto poderia ter sido também uma CONEXÃO INTERNA utilizando: ON mc.id\_contato = ed.ed\_id\_contato.*

```
CREATE VIEW escritor_tecnico_emprego AS
SELECT cargo, salario, descricao, cep
FROM lista_emploies
WHERE cargo = 'Escritor Técnico' ;
```



Ah haa, fácil! Mas  
como de fato utilizo  
as views que criei?

 **PODER DO CÉREBRO**

Como você acha que um comando SQL que utiliza uma view se parece?

## visualizando suas views

Consideraremos a view web\_designers que acabamos de criar:

```
CREATE VIEW web_designers AS
```

```
SELECT mc.primeiro_nome, mc.sobrenome, mc.fone, mc.email
FROM meus_contatos mc
NATURAL JOIN emprego_desejado ed
WHERE ed.cargo = 'Programador Web' ;
```

*Lembre-se: somos permitidos a deixar de fora a palavra-chave AS.*

Para ver o que há nela, simplesmente a tratamos como se fosse uma tabela. Nós podemos utilizar o SELECT:

```
SELECT * FROM web_designers;
```

*Aqui está o nome da  
nossa view.*

O resultado é:

primeiro_nome	sobrenome	fone	email
John	Martinez	5559872	jm@someemail.com
Samantha	Hoffman	5556948	sammy@someemail.com
Todd	Hertz	5557888	tod@someemail.com
Fred	McDougal	5557744	fm@someemail.com

*E assim por diante, até que todas as linhas que  
se ajustem a Web Designer sejam listadas.*

## O que sua view está realmente fazendo

Quando for utilizar sua view de fato em uma consulta, ela estará se comportando como se fosse uma subconsulta. Aqui está o que nossa subconsulta que acabamos de utilizar está realmente dizendo ao Sistema SQL para ser feito:

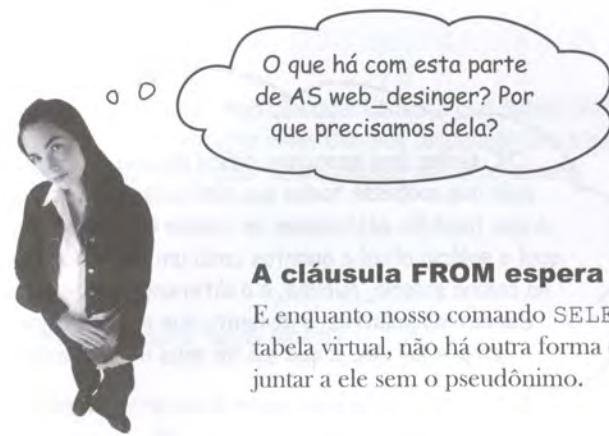
```
SELECT * FROM web_designers;
```

Isto quer dizer, “Selecione tudo da subconsulta que retorne o primeiro nome, sobrenome, telefone e email de todas as pessoas de meus\_contatos que estão procurando por um emprego como web designer”.

```
SELECT * FROM web_designers;
(SELECT mc.primeiro_nome, mc.sobrenome, mc.fone, mc.email
FROM meus_contatos mc
NATURAL JOIN emprego_desejado ed
WHERE ed.cargo = 'Programador Web') AS web_designers;
```

*Aqui está o que utilizamos  
em nossa view.*

*Estamos dando a nossa subconsulta  
um pseudônimo para que a consulta  
trate como uma tabela.*



### A cláusula FROM espera por uma tabela.

E enquanto nosso comando SELECT resulta em uma tabela virtual, não há outra forma do SQL poder se juntar a ele sem o pseudônimo.

## que é uma view

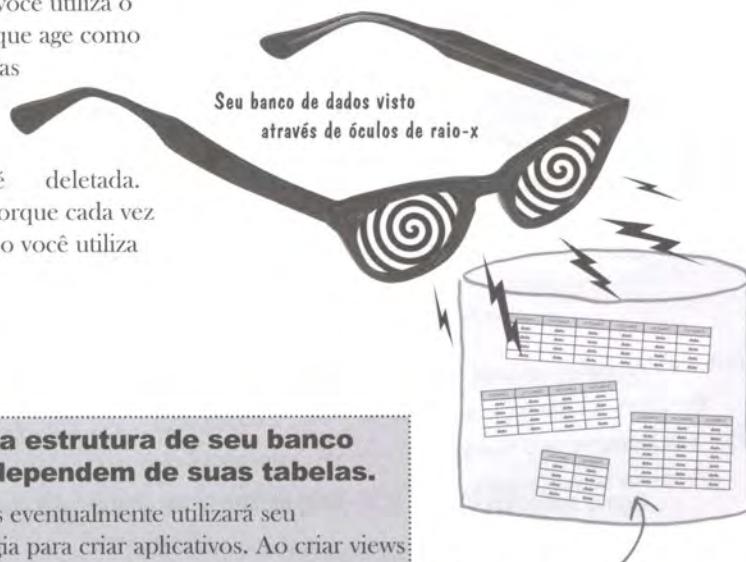
VIEW é basicamente uma tabela que só existe quando você utiliza o em uma consulta. É considerada uma tabela virtual porque age como tabela, e as mesmas operações que podem ser executadas na tabela podem ser executadas em uma view.

tabela virtual não permanece no banco de dados. Criada quando utilizamos o banco de dados e depois é deletada. A VIEW é a única coisa que permanece. Isto é bom porque cada vez linhas forem inseridas em um banco de dados, quando você utiliza VIEW e ela verá a nova informação.

### que as views são boas para seu banco de dados

#### **Você pode evitar que as alterações da estrutura de seu banco de dados travem as aplicações que dependem de suas tabelas.**

Nós ainda não falamos sobre isto nesse livro, mas eventualmente utilizará seu conhecimento de SQL e usará com outra tecnologia para criar aplicativos. Ao criar views nos seus dados, você será capaz de alterar a estrutura base de sua tabela, mas criar views que limitarão a forma que sua tabela costumava ser, para que não tenha que alterar seu aplicativo que utiliza destes dados.



Essas tabelas só existem porque usamos VIEW nossas consultas.

#### **Views tornam sua vida mais fácil ao simplificar suas consultas complexas em um simples comando.**

Você não terá que criar conexões complicadas e repetidas subconsultas quando se cria uma view. Sua view esconde a complexidade de sua consulta base. E quando você vincula sua SQL a um PHP ou outra linguagem de programação, sua view será muito mais fácil de adicionar ao seu código. Você está utilizando o código simplificado da view e não o grande e complexo código da consulta, cheia de conexões. Simplicidade quer dizer que há menos chances de erros de digitação, e seu código será muito mais fácil de ser lido.

#### **Você pode criar views que escondem informações que não são necessárias ao usuário.**

Considere a eventual necessidade de se adicionar tabelas ao banco de dados gregs\_list que contém informações de cartões de crédito. Você pode criar uma view para indicar que alguém tem um cartão em arquivo, sem revelar os detalhes do cartão. Pode ainda permitir que empregados vejam apenas a informação que eles precisam ver, enquanto mantém informações sensíveis escondidas.



OK, tenho uma pergunta difícil para você. Eu poderia criar uma view que exibisse todos que estivessem na tabela emprego\_atual e que também estivessem na tabela emprego\_desejado, bem como qual o salário atual e quantos cada um pretende ganhar, baseando-se na coluna salario\_minimo, e a diferença entre estas duas quantias? Em outras palavras, o aumento que eles desejam para mudar de emprego? Ah...e que me dê seus nomes, emails e telefone.



### Exercícios

Quais são as tabelas que precisarão estar nesta tabela?

.....

Quais as colunas de cada tabela que podem ser usadas para descobrir o aumento?

.....

Como podemos utilizar SQL para, de fato, criar uma coluna chamada 'aumento' em nossos resultados?

.....

Escreva a consulta de Frank:

.....

.....

.....

.....

.....

Click: este slide deve ser usado com slides conexões mais frias para elas.



## Solução dos exercícios

Quais são as tabelas que precisarão estar nesta tabela?

*emprego\_atual, emprego\_desejado e meus\_contatos*

Quais as colunas de cada tabela que podem ser usadas para descobrir o aumento?

*A coluna salario em emprego\_atual e a coluna salario\_minimo em emprego\_desejado.*

Como podemos utilizar SQL para, de fato, criar uma coluna chamada 'aumento' em nossos resultados?

*Subtraia salario\_atual de salario\_minimo e dê um pseudônimo.*

Escreva a consulta de Frank:

```
CREATE VIEW aumento_salario AS
SELECT mc.primeiro_nome, mc.sobrenome, mc.email, mc.fone, ea.id_contato, ea.salario, ed.salario_
minimo, ed.salario_minimo - ea.salario AS aumento
FROM emprego_atual ea
INNER JOIN emprego_desejado ed
INNER JOIN meus_contatos mc
WHERE ea.id_contato = ed.id_contato
AND ea.id_contato = mc.id_contato;
```

*Aqui nós criamos nossa nova view chamada de aumento\_salario*

*Depois de criarmos a view, o resto da consulta utiliza CONEXÕES INTERNAS para puxar os dados de três tabelas. Também utilizamos uma pequena operação matemática para criar nossa nova coluna aumento.*

*Esta subtrai o salário que eles querem do salário que eles recebem agora e utiliza um pseudônimo para chamar os resultados de aumento.*

**É uma consulta enorme, mas agora tudo o que Frank precisa fazer é digitar.**

**SELECT \* FROM aumento\_salario;**

**para visualizar esta informação.**

Aponte seu lápis —————

Se ele executar o SELECT da página 379 utilizando a nova view aumento\_salario, como Frank poderá ordenar os resultados alfabeticamente pelo sobrenome?

→ Respostas na página 399.

## Inserindo, atualizando e deletando com views

Você pode fazer mais do que SELECT informações de suas tabelas com uma view. Em alguns casos pode UPDATE, INSERT e DELETE seus dados da mesma maneira.



Então posso criar uma view que me permitirá alterar de fato o que está na minha tabela?

**Você pode, mas não vale a pena o trabalho.**

**Se sua view utilizar valores agregados** (como SUM, COUNT e AVG), você não está apto a utilizá-la para alterar os dados. Também se suas views contiverem GROUP BY, DISTINCT ou HAVING, elas também não alterarão os dados.

Na maioria das vezes será mais fácil utilizar INSERT, UPDATE e DELETE da forma antiga, mas mostraremos um exemplo de como alterar seus dados com uma view na próxima página.

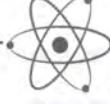
## O segredo é fingir que a view é uma tabela de verdade

Vamos fazer uma view a partir de uma nova tabela chamada cofrinho. Esta tabela contém moedas que estamos juntando. Existe um ID para cada moeda; uma coluna descrição indica se é uma moeda de Um, Cinco, Dez ou Vinte e cinco centavos; e o ano em que aquela moeda foi cunhada.

```
CREATE TABLE cofrinho
(
    id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    moeda CHAR(1) NOT NULL,
    ano_moeda CHAR(4)
)
```

E aqui estão os dados atualmente inseridos na tabela cofrinho:

id	moeda	ano_moeda
1	V	1950
2	U	1972
3	C	2005
4	V	1999
5	V	1981
6	D	1940
7	V	1980
8	U	2001
9	D	1926
10	U	1999

 **PODER DO CÉREBRO**

Como será a tabela resultados quando executarmos esta tabela?

```
SELECT * FROM c_vinteecinco;
```

Vamos escrever uma view que nos exiba apenas as moedas de Vinte e cinco centavos:

```
CREATE VIEW AS c_vinteecinco
SELECT * FROM cofrinho
WHERE moeda = 'V' ;
```



## Exercícios

Tente isto em casa. Crie a tabela cofrinho e as views c\_vinteecinco e c\_dez utilizando as consultas exibidas abaixo.

```
INSERT INTO cofrinho VALUES ('', 'V', 1950), ('', 'U', 1972), ('', 'C', 2005),  
('', 'V', 1999), ('', 'V', 1981), ('', 'D', 1940), ('', 'V', 1980), ('', 'U', 2001), ('', 'D',  
1926), ('', 'U', 1999);
```

```
CREATE VIEW c_vinteecinco AS SELECT * FROM cofrinho WHERE coin = 'V';
```

```
CREATE VIEW c_dez AS SELECT * FROM cofrinho WHERE coin = 'D' WITH CHECK OPTION;
```

Escreva o que acontece quando você executa cada uma destas consultas: INSERT, DELETE e UPDATE. Ao final deste exercício, rascunhe a tabela final de cofrinho.

Tente descobrir o que isto aqui faz na medida em que avança neste exercício.

```
INSERT INTO c_vinteecinco VALUES ('', 'V', 1993);
```

```
INSERT INTO c vinteecinco VALUES ('', 'D', 1942);
```

```
INSERT INTO c_dez VALUES ('', 'v', 2005);
```

```
DELETE FROM c_vintecinco WHERE moeda = 'N' OR moeda = 'U' OR moeda = 'D';
```

```
UPDATE c vinteecinco SET moeda = 'V' WHERE moeda = 'U';
```



## Solução dos Exercícios

Tente isto em casa. Crie a tabela cofrinho e as views c\_vinteeccinco e c\_dez utilizando as consultas exibidas abaixo.

```
INSERT INTO cofrinho VALUES ('V', 1950), ('U', 1972), ('C', 2005),
('V', 1999), ('V', 1981), ('D', 1940), ('V', 1980), ('U', 2001), ('D',
1926), ('U', 1999);
```

```
CREATE VIEW c_vinteeccinco AS SELECT * FROM cofrinho WHERE coin = 'V';
```

```
CREATE VIEW c_dez AS SELECT * FROM cofrinho WHERE coin = 'D' WITH CHECK OPTION;
```

Escreva o que acontece quando você executa cada uma destas consultas: INSERT, DELETE e UPDATE. Ao final deste exercício, rascunhe a tabela final de cofrinho.

```
INSERT INTO c_vinteeccinco VALUES ('V', 1993);
```

*Esta consulta será executada apropriadamente.*

```
INSERT INTO c_vinteeccinco VALUES ('D', 1942);
```

*Esta insere um novo valor na tabela, embora você tenha pensado que ela não poderia fazê-lo por causa da cláusula WHERE.*

```
INSERT INTO c_dez VALUES ('V', 2005);
```

*Esta exibe um erro por causa das cláusulas CHECK OPTION. Ela faz com que os dados inseridos em uma view sejam verificados mediante uma cláusula WHERE antes de ser autorizada a ser inserida.*

```
DELETE FROM c_vinteeccinco WHERE moeda = 'N' OR moeda = 'U' OR moeda = 'D';
```

*Esta não faz nada à tabela porque ela só procurar por moedas com resultados para coluna moeda = V*

```
UPDATE c_vinteeccinco SET moeda = 'V' WHERE moeda = 'U';
```

*Esta não faz nada à tabela porque nenhum dos valores de moedas = V são retornadas pela view c\_vinteeccinco.*

*A tabela final se parece com isso:*

id	moeda	moeda_ano
1	V	1950
2	U	1972
3	C	2005
4	V	1999
5	V	1981
6	D	1940
7	V	1980
8	U	2001
9	D	1926
10	U	1999
11	V	1993
12	D	1942

## View com CHECK OPTION

O CHECK OPTION adicionado a sua view diz ao Sistema SQL para checar se o comando que você tenta utilizar INSERT ou DELETE para verificar se o comando é permitido de acordo com a cláusula WHERE na sua view. Então, como o CHECK OPTION afeta seus comandos INSERT e UPDATE?

Quando você usou CHECK OPTION no exercício anterior, seus dados seriam salvados no seu comando INSERT se não tivessem satisfeito a condição WHERE na view c\_dez. Se usar um UPDATE também terá um erro:

```
UPDATE c_dez SET moedas = 'x' ;
```

A condição WHERE em c\_dez não foi satisfeita por 'x' então nada foi atualizado.



Não seria possível utilizar views com CHECK OPTION para criar algo como uma CHECK CONSTRAINT se estiver utilizando o MySQL?

Em MySQL, você pode imitar uma CHECK CONSTRAINT utilizando uma CHECK OPTION

**Sim, suas consultas podem precisamente espelhar o que está na tabela, mas force os comandos INSERT para cumprirem as cláusulas WHERE.**

Por exemplo, com o nosso problema de sexo anteriormente neste capítulo, nós poderíamos criar uma view da tabela my\_contacts que Jim poderia usar para atualizar my\_contacts. Isto poderia simplesmente provocar um erro todas as vezes que ele tentar colocar X na tabela de sexo.



### Poder do Cérebro

Como poderíamos criar uma view para meus\_contatos que forçaria Jim a inserir tanto 'M' ou 'F' para o campo sexo?

## Sua view pode ser atualizável, se....

A tabela cofrinho, ambas as views que criamos são views atualizáveis. Uma view **atualizável** é aquela que permite alterar tabelas bases. O ponto importante aqui é que uma view atualizável inclui todas as colunas NOT NULL das tabelas a qual faz referência. Desta forma, quando você utiliza o INSERT utilizando uma view, pode ter certeza que terá um valor para cada coluna que for pedida a você para constar um valor inserido.

Basicamente, isto quer dizer que INSERT, UPDATE e DELETE podem ser usados em as views que criamos. Desde que a view retorne quaisquer colunas da tabela que sejam not null, a view poderá inserir valores apropriados para a tabela.

Existem também as views não-atualizáveis. Uma **view não-atualizável** é aquela que não inclui todas as colunas NOT NULL. Com exceção de criá-las e apagá-las, a única coisa que se pode fazer com uma view não-atualizável é utilizar o SELECT.

CHECK OPTION checa cada consulta que você tenta utilizar INSERT ou DELETE para verificar se o comando é permitido de acordo com a cláusula WHERE na sua view.

Uma view atualizável inclui todas as colunas NOT NULL das tabelas a qual faz referência

Com exceção de utilizar o CHECK OPTION, não vejo objetivo de usar view para utilizar o INSERT.



### É verdade, você não vai utilizar views com freqüência para utilizar o INSERT, UPDATE ou DELETE.

Enquanto há usos válidos, tais como forçar a integridade dos dados com MySQL, geralmente é mais fácil usar a própria tabela para utilizar o INSERT, UPDANTE ou DELETE. Um INSERT em uma view se mostra útil quando a view revela apenas uma coluna e as demais colunas são designadas com valores NULL ou valores padrões. Neste caso, então o INSERT faz sentido. Você pode também adicionar uma cláusula WHERE para sua view que restringirá o que pode inserir, ajudando-o a imitar a CHECK CONSTRAINING do MySQL.

Para deixar as coisas ainda mais confusas, você pode somente atualizar views que não contenham operadores agregados como SUM, COUNT e AVG e operadores como BETWEEN, HAVING, IN e NOT IN.

## Quando você tiver terminado com sua view

Quando não precisar mais de suas views, limpe-as utilizando um comando DROP VIEW. É tão simples quanto abaixo:

```
DROP VIEW c_dez;
```

---

não existem  
Perguntas Idiotas

---

P: Há alguma maneira de visualizar as views que você já criou?

R: As views são exibidas da mesma forma que as tabelas no seu banco de dados. Você pode usar o comando SHOW TABLES para ver todas as views e tabelas. E da mesma forma que a tabela, pode utilizar o comando DESC para ver sua estrutura.

P: O que acontece se eu eliminar uma tabela que possui uma view?

R: Depende. Alguns Sistemas SQL ainda permitirão que utilize a view, mas não retornarão dados. MySQL não permitirá que você apague uma view, a não ser que a tabela na qual ela se baseia exista, embora você possa eliminar uma tabela que seja participante em uma view. Outros Sistemas têm diferentes comportamentos. É uma boa idéia experimentar com o seu Sistema para ver o que acontece. Em geral, é melhor eliminar o view antes de eliminar a tabela na qual ela está baseada.

P: Eu vejo o quanto as CHECK CONSTRAINTS e as VIEWS são úteis para ajudar quando mais de uma pessoa está tentando fazer coisas no banco de dados. Mas o que acontece quando duas pessoas estão tentando alterar a mesma coluna na mesma hora?

R: Para isto, deveríamos falar sobre transações. Mas primeiro, Sra. Humphries precisa conseguir algum dinheiro.

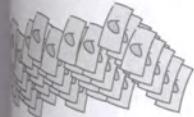
**CHECK CONSTRAINTS**  
e views ajudam a manter  
o controle quando se têm  
múltiplos usuários.

## ... quando coisas ruins acontecem com bons bancos de dados

Mrs. Humphries quer transferir 1000 reais da sua conta corrente para sua conta poupança. Ela vai ao caixa eletrônico...

...para checar o saldo de sua conta corrente e poupança.

0 reais em sua conta corrente



30 reais em sua conta poupança



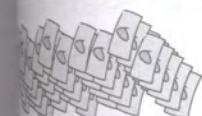
...e aperta o botão.

**TRANSFERIR 1000 REAIS DA CONTA CORRENTE PARA POUPANÇA**

...e aperta o botão.

CONTA CORRENTE

POUPANÇA



Caixa eletrônico emite um som baixo. A eletricidade acabou.



### A eletricidade volta.

Ela checa os saldos de sua conta corrente e poupança.

0 REAIS NA CONTA CORRENTE

30 REAIS NA POUPANÇA



**Onde, oh onde foi parar o dinheiro da Sra. Humphries?**

# O que aconteceu dentro do caixa eletrônico?



Foi aí que a energia acabou.

**CAIXA:** LA LA LA LA LA.

**CAIXA:** OLHE, E A SRA. ETHEL P. HUMPHRIES. OI SRA. ETHEL P. HUMPHRIES (ID\_CONTA 38221 )

**SRA. Humphries:** Diga-me quanto dinheiro tenho.

**CAIXA:** Pensando ( SELECT SALDO FROM CORRENTE WHERE ID\_CONTA = 38221  
SELECT SALDO FROM POUPANCA WHERE ID\_CONTA = 38221 : )  
ENTÃO SAO 1000 NA CORRENTE, 30 NA POUPANÇA.

**SRA. Humphries:** Transfira estes 1000 reais para a conta poupança.

**CAIXA:** ESTA É UMA ORDEM DIFÍCIL, SRA. HUMPHRIES, MAS LÁ VAI : (SALDO\_POUPANCA > 1000. ENTAO ELA TEM DINHEIRO SUFICIENTE ) [ REMOVE 1000 DA CONTA CORRENTE ]

( INSIRA BEEEEEEP.... )

**CAIXA:**

**CAIXA:**

**CAIXA:** ZZZZZZZZ

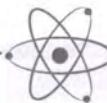
**CAIXA:** QUE SONO...

**CAIXA:** OLHE, E A SRA. ETHEL P. HUMPHRIES. OI SRA. ETHEL P. HUMPHRIES (ID\_CONTA 38221 )

**SRA. Humphries:** Diga-me quanto dinheiro eu tenho.

**CAIXA:** Pensando ( SELECT SALDO FROM CORRENTE WHERE ID\_CONTA = 38221 : SELECT SALDO FROM POUPANCA WHERE ID\_CONTA = 38221 ; )  
ENTÃO SAO 0 NA CORRENTE, 30 NA POUPANÇA.

**CAIXA:** AI!!! A SRA ESTA BATENDO NA MINHA TELA. ATÉ LOGO, SRA. ETHEL P. HUMPHRIES !



## PODER DO CÉREBRO

Como poderíamos ter prevenido o CAIXA ELETRÔNICO de esquecer quanto à transação da Sra. Humphries?

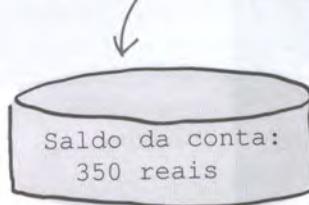
Enquanto isso, do outro lado da cidade...

## Problemas no CAIXA ELETRÔNICO

Mary e John possuem uma conta conjunta. Na sexta-feira, eles se encontraram em dois caixas eletrônicos ao mesmo tempo. Ambos tentaram sacar 300 reais.



Aqui está o banco de dados mantendo a contagem de quanto há na conta conjunta de John e Mary.



**John:** OH! É VOCÊ NOVAMENTE JOHN. VOCÊ ACHA QUE EU FEITA DE DINHEIRO?

**John:** Qual o meu saldo?

**John:** Pensando ( SELECT SALDO\_CORRENTE FROM CONTAS ;  
350 REAIS )

**John:** 300 reais

**John:** É SÓ PRA ISSO QUE VOCÊ ACHA QUE SIRVO.

**John:** DAR DINHEIRO. APENAS ME USA E DEPOIS ME IGNORA.

**John:** ( SALDO\_CORRENTE > 300. ELE TEM DINHEIRO SUFICIENTE )

**John:** ( REMOVE 300 FROM CORRENTE )

350 reais      350 reais

**John:** ( SUBTRAIA 300 DO SALDO\_CORRENTE )

50 reais

-250 reais

John pega o dinheiro e corre.

Foi aqui que tudo  
deu errado...

**Mary:** VOCÊ NUNCA LIGA E NUNCA ESCREVE.

JOHN.

**CAIXA:** MARY, OLÁ.

**Mary:** Qual o meu saldo?

**CAIXA:** Pensando ( SELECT SALDO\_CORRENTE FROM CONTAS : ) 350 REAIS

**RING RING**

*Mary mexe em sua bolsa, procurando seu celular.*

**Mary:** 300 reais.

**CAIXA:** PODE CRER!

( SALDO\_CORRENTE > 300. ELA TEM DINHEIRO SUFICIENTE )

( REMOVE 300 FROM CORRENTE )

( SUBTRAIA 300 DO SALDO\_CORRENTE )

**CAIXA:** VOCÊ ESTÁ COM O SALDO BASTANTE NEGATIVO.



Não seria um sonho se uma série de comandos SQL pudessem ser executados como um grupo, tudo de uma vez, e se algo der errado, e se algo desse errado, ele retrocedesse deixando tudo como se o comando nunca tivesse sido executado? Mas isto é apenas um sonho....

## Isto não é um sonho, é uma transação

Uma **transação** é um conjunto de comandos SQL que alcançam um trabalho único. No caso da Sra. Humphries, uma transação consistiria de todos os comandos SQL para poder remover o dinheiro de sua conta corrente para sua conta poupança:

*Estas três ações formam uma unidade única de trabalho. Aqui está a transação.*

**Se o saldo corrente  $\geq 1000$**

**Subtraia 1000 do saldo corrente**

**Adicione 1000 no saldo poupança**

John e Mary estavam ambos tentando fazer a **mesma transação** ao mesmo tempo:

*John e Mary estavam ambos tentando sacar 300 reais ao mesmo tempo.*

**Se o saldo corrente  $\geq 300$**

**Subtraia 300 do saldo corrente**

**Distribua 300 reais**

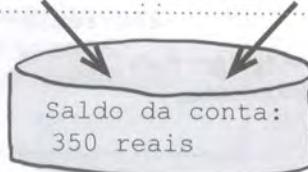
*A transação de John no Caixa Eletrônico do Banco Esquerdo*

**Se o saldo corrente  $\geq 300$**

**Subtraia 300 do saldo corrente**

**Distribua 300 reais**

*A transação de Mary no Caixa Eletrônico do Primeiro Banco Nacional.*



No caso de John e Mary, o Caixa Eletrônico do Primeiro Banco Nacional não deveria ter permitido que a conta fosse tocada, ainda que fosse para consultar o saldo, até que o Caixa Eletrônico do Banco Esquerdo tivesse terminado com a transação, desta forma destrancando-o.

Durante uma transação, se todos os passos não puderem ser completados sem interferência, nenhum deles deverá ser realizado.

## este clássico ACID

ajudar a decidir quais passos poderão ser considerados parte de uma transação, lembre-se do acrônimo **ACID**. Há quatro características que devem ser verdadeiras antes de podermos chamar um conjunto de comandos SQL de transação.



### ACID: ATOMICIDADE

Todas as partes da transação devem ser completadas ou nenhuma delas será completada. Você pode executar parte da transação. Os reais da Sra. Humphries foram para lugar nenhum num piscar de olhos pela falta de energia, pois apenas parte da transação foi executada.



### ACID: CONSISTÊNCIA

Uma transação completa deixa o banco de dados consistente ao final da transação. Ao final de ambas as transações de reais, o dinheiro estava constando no saldo novamente. No primeiro caso, ele foi transferido da poupança; no segundo ele foi traduzido em dinheiro, mas não ficou um real sequer faltando.



### ACID: ISOLAMENTO

Isolamento quer dizer que cada transação deve ter uma visão constante do banco de dados, sem levar em consideração qualquer outra transação. Ela não devia estar apta a ver o saldo, ou deveria ter visto alguma mensagem do tipo “transação em progresso”.



### ACID: DURABILIDADE

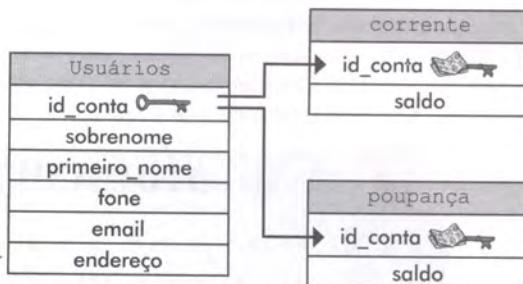
Depois da transação, o banco de dados precisa salvar os dados corretamente e protegê-lo da falta de energia ou outras ameaças. Isto geralmente é gerenciado através dos registros de transações salvas em local diverso do que o banco de dados principal. Se um registro da transação da Sra. Humphries tivesse sido deixado em algum lugar, então ela poderia ter recebido seus 1.000 reais de volta.

# SQL ajuda a gerenciar suas transações

Vamos considerar um banco de dados de um banco bastante simples. Nossa base de dados consiste em uma tabela com o nome dos correntistas, uma tabela de conta corrente e uma tabela de conta poupança

*Provavelmente, há muito mais colunas aqui, mas você já entendeu a ideia.*

Temos três ferramentas SQL para transações para ajudar a nos manter seguros.



**START TRANSACTION;**

**START TRANSACTION rastreia todo o SQL** que segue até você inserir o comando COMMIT ou ROLLBACK.

Aqui é onde o Sistema SQL começa a rastrear seu código.



**COMMIT;**

Este comando confirma todo o seu código uma vez que esteja feliz com ele.

Se você está com todos os seus comandos no devido lugar e tudo parece bom, digite COMMIT para torná-lo permanente.

Uma vez estando feliz com seu código, você pode confirmá-lo para seu banco de dados...

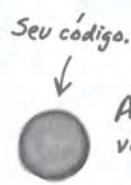


**ROLLBACK;**

Este leva você de volta para onde começou sua transação.

Se algo não estiver muito certo, ROLLBACK reverte tudo até a forma como estava **antes** de você START TRANSACTION (começar a transação).

...ou você pode ROLLBACK (retroceder) para como as coisas estavam antes de você começar.



Onde você começou.

Nenhuma alteração ocorrerá até você digitar COMMIT

## que deveria ter acontecido dentro do Caixa eletrônico



**CAIXA:** LA LA LA LA LA.

**CAIXA:** OLHE, E A SRA. ETHEL P. HUMPHRIES. OI, SRA. ETHEL P. HUMPHRIES!  
(ID\_CONTA 38221)

**SRA. HUMPHRIES:** DIGA-ME QUANTO DINHEIRO EU TENHO.

**CAIXA:** Pensando (SELECT SALDO FROM CORRENTE WHERE ID\_CONTA = 38221 ; )

SELECT SALDO FROM POUPANÇA WHERE ID\_CONTA = 38221 ; )

ENTAO SAO 1000 NA CORRENTE, 30 NA POUPANCA.

**SRA. HUMPHRIES:** Transfira ESTES 1000 reais para a Conta Poupança.

**CAIXA:** É UM PEDIDO BASTANTE DIFÍCIL, SRA. HUMPHRIES, MAS AQUI VAI :  
(START TRANSACTION ;  
SELECT SALDO FROM CORRENTE WHERE ID\_DA\_CONTA = 38221 ; )

**CAIXA:** ELA TEM 1000 NA CONTA CORRENTE, PORTANTO VOU CONTINUAR.

**CAIXA:** ( UPDATE CONTA\_CORRENTE SET SALDO = SALDO - 1000 WHERE  
ID\_DA\_CONTA = 38221 ; )

→ (INSIRA BEEEEEEP....)

*Agui a energia  
acabou.*

CAIXA ELETRONICO SEM ENERGIA. ROLLBACK ;

**CAIXA:**

**CAIXA:**

**CAIXA:** ZZZZZZZZ

**CAIXA:** QUE SONO...

**CAIXA:** OLHE, E A SRA. ETHEL P. HUMPHRIES. OI SRA. ETHEL P. HUMPHRIES  
(ID\_CONTA 38221 )

**SRA. HUMPHRIES:** DIGA-ME QUANTO DINHEIRO EU TENHO.

**CAIXA:** Pensando (SELECT SALDO FROM CORRENTE WHERE ID\_CONTA =  
38221;

SELECT SALDO FROM POUPANCA WHERE ID\_CONTA = 38221 ; )

*Gracas ao ROLLBACK  
o comando COMMIT  
nunca foi inserido e  
nada foi alterado.*

→ ENTÃO SÃO 1000 NA CORRENTE, 30 NA POUPANÇA.

# Como fazer as transações funcionarem com MySQL

Antes de utilizar uma transação do MySQL, você precisa usar o **mecanismo de armazenamento** correto.

O mecanismo de armazenamento é a estrutura por detrás das cortinas que armazena todos os dados e estruturas do seu banco de dados. Alguns tipos permitem mecanismos de armazenamento, outros não.

Pense novamente no Capítulo 4 quando você viu o

`SHOW CREATE TABLE meus_contatos;`

Use a Cabeça SQL

## tabela Mostre-me o dinheiro

E se você utilizar o comando `DESCRIBE meus_contatos` para verificar o código que você utilizou quando criou a tabela? Você verá algo mais ou menos assim:

Column	Type	Null	Key	Default	Extra
sobrenome	varchar(30)	YES		NULL	
primeiro_nome	varchar(20)	YES		NULL	
email	varchar(50)	YES		NULL	
sexo	char(1)	YES		NULL	
aniversario	date	YES		NULL	
profissao	varchar(50)	YES		NULL	
local	varchar(50)	YES		NULL	
estado_civil	varchar(20)	YES		NULL	
interesses	varchar(100)	YES		NULL	
procura	varchar(100)	YES		NULL	

Mas o que nós realmente queremos ver é o código `CREATE` aqui e não os campos da tabela para que possamos imaginar o que teríamos feito lá no inicio sem ter que escrever o comando `CREATE` todo novamente.

O comando `SHOW CREATE_TABLE` irá retornar o comando `CREATE TABLE` que pode recriar sua tabela, com exceção dos dados nela inseridos. Desta forma, você sempre pode ver como a tabela que está olhando pode ser criada. Experimente:

`SHOW CREATE TABLE meus_contatos;`

## Comando para economia de tempo

Dê uma olhada no comando que utilizamos para criar a tabela, e o código abaixo que o comando `SHOW CREATE TABLE meus_contatos` lhe deu. Eles não são idênticos, mas se colar o texto abaixo no seu comando `CREATE TABLE`, o resultado final será o mesmo. Você não precisa remover os acertos graves ou configurações de dados, mas fica mais praticado se o fizer.

```
CREATE TABLE `meus_contatos` (
  `sobrenome` varchar(30) default NULL,
  `primeiro_nome` varchar(20) default NULL,
  `email` varchar(50) default NULL,
  `sexo` char (1) default NULL,
  `aniversario` date default NULL,
  `profissao` varchar(50) default NULL,
  `local` varchar(50) default NULL,
  `estado_civil` varchar(20) default NULL,
  `interesses` varchar(100) default NULL,
  `procura` varchar(100) default NULL,
  ENGINE=MyISAM DEFAULT CHARSET=latin1
)
Ds sinal aberto e fechando os nomes das colunas e o nome da
tabela não chamados de acentos graves. Esses aparecem grande
excertuadas o comando SHOW CREATE TABLE.
A não ser que digamos
é uma boa ideia
especificar se uma
coluna contém
não valores NULL
seu NULL como padrão.
Voçê não precisa se preocupar quanto
à ultima linha do texto após fechar
os parênteses. Ia especificar como
as linhas serão armazenadas e qual
o conjunto de caracteres a ser
utilizada. A definição padrão está
atômica, por enquanto.
A não ser que tenha
diferenciado a tabela original,
voçê terá que dar um nome
diferente.
```

150 Capítulo 4

*Desta vez nos importamos quanto ao mecanismo de busca.*

*Você não precisa se preocupar em relação à última linha de texto depois do fechamento do parêntesis. Ela especifica como os dados serão armazenados e qual conjunto de caracteres deve ser usado. Por hora, as configurações default são adequadas.*

**Você precisa ter certeza de que seu mecanismo de armazenamento seja tanto BDB ou InnoDB, as duas opções que suportam transações.**



**InnoDB e DBD são duas possíveis formas que seu Sistema SQL pode armazenar seus dados por detrás das cortinas.**

Eles são chamados de mecanismo de armazenamento, e utilizando qualquer um desses dois tipos você terá certeza que o Sistema suportará transações. Consulte um manual para maiores diferenças entre os mecanismos de armazenamento que o MySQL oferece.

Para nossos propósitos atuais, não importa qual irá utilizar. Para alterar seu mecanismo utilize a seguinte sintaxe:

`ALTER TABLE sua_tabela = InnoDB;`

## pra tentar você mesmo

da que tenhamos trocado todas as moedas de centavo no cofrinho por moedas de Vinte e Centavos.

No código abaixo você mesmo na tabela cofrinho que criamos anteriormente neste capítulo. Decidimos utilizar um ROLLBACK porque não queremos fazer as alterações:

```
START TRANSACTION;
SELECT * FROM cofrinho;
UPDATE cofrinho set moeda = 'V' where moeda= 'U';
SELECT * FROM cofrinho; ← Agora você vê as alterações...
ROLLBACK; ← Mudamos de idéia.
SELECT * FROM cofrinho; ← ...e agora não
```

Na segunda vez que utilizaremos o COMMIT porque queremos com as alterações.

```
START TRANSACTION;
SELECT * FROM cofrinho;
UPDATE cofrinho set moeda = 'V' where moeda= 'U';
SELECT * FROM cofrinho; ← Agora você vê as alterações...
COMMIT; ← Faz as alterações serem permanentes.
SELECT * FROM cofrinho; ← ...e ainda vê as alterações.
```



Aponte seu lápis—

Preencha as lacunas dos conteúdos após as transações.  
Aqui está como ela se encontra atualmente:

```
START TRANSACTION;
UPDATE cofrinho set moeda = 'V' where moeda = 'U'
AND moeda_ano < 1970;
COMMIT;
```

cofrinho		
id	moeda	moeda_ano
1	V	1950
2	U	1972
3	C	2005
4	V	1999

id	moeda	moeda_ano
1		
2		
3		
4		

```
START TRANSACTION;
UPDATE cofrinho set moeda = 'C' where moeda = 'V';
ROLLBACK;
```

id	moeda	moeda_ano
1		
2		
3		
4		

```
START TRANSACTION;
UPDATE cofrinho set moeda = 'V' where moeda = 'C'
AND moeda_ano > 1950;
ROLLBACK;
```

id	moeda	moeda_ano
1		
2		
3		
4		

```
START TRANSACTION;
UPDATE cofrinho set moeda = 'D' where moeda = 'V'
AND moeda_ano > 1980;
COMMIT;
```

id	moeda	moeda_ano
1		
2		
3		
4		

```
START TRANSACTION;
UPDATE cofrinho set moeda = 'U' where moeda = 'C'
AND moeda_ano > 1970;
COMMIT;
```

id	moeda	moeda_ano
1		
2		
3		
4		

→ Respostas na página 400.

## não existem Perguntas Idiotas

P: Você deve começar com o comando START TRANSACTION, ou COMMIT e ROLLBACK funcionam sem ele?

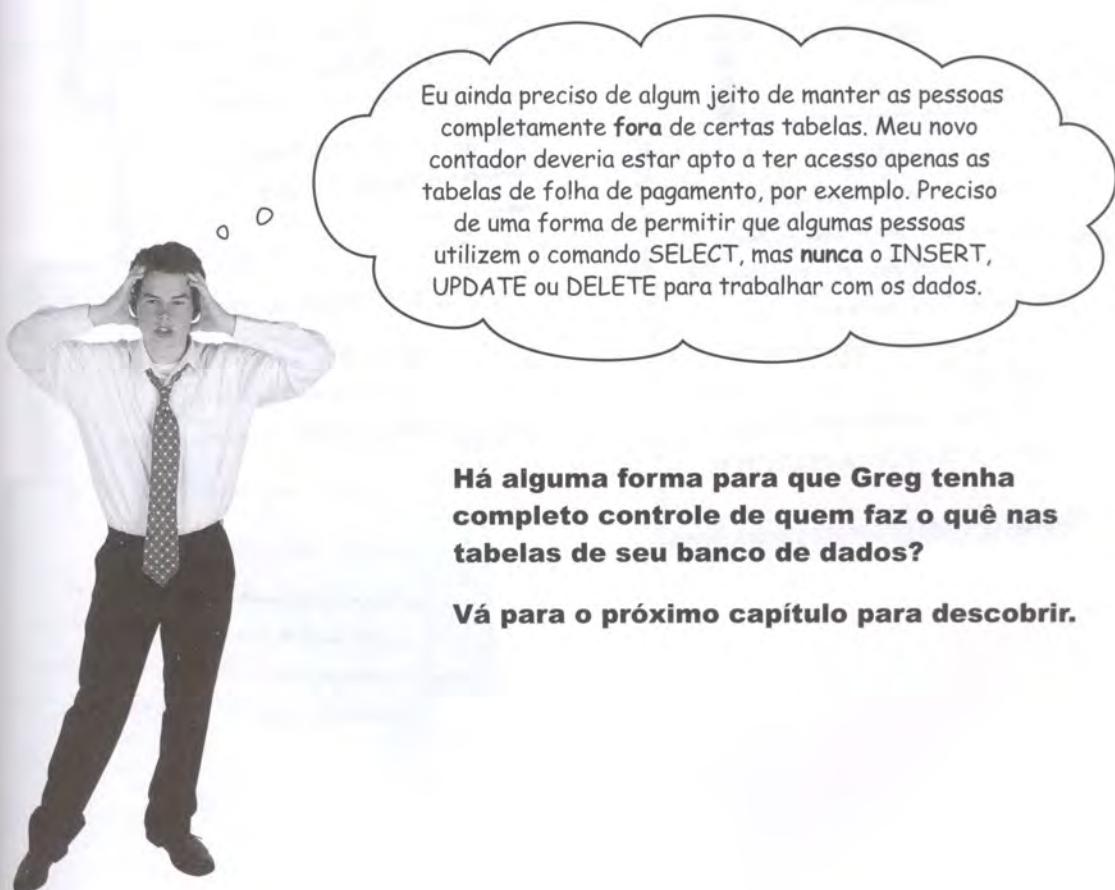
R: Você deve dizer ao seu Sistema SQL que está iniciando uma transação com START TRANSACTION. Ele mantém registrado quando sua transação começou para saber o quanto retroceder para desfazer tudo.

P: Posso apenas utilizar START TRANSACTION para que eu experimente algumas consultas?

R: Você pode e deve. É uma ótima maneira de praticar consultas que alteram dados em suas tabelas sem alterar permanentemente as tabelas se tiver feito algo errado. Apenas certifique-se que utilizou COMMIT ou ROLLBACK quando tiver terminado.

P: Porque devo me preocupar com o COMMIT e ROLLBACK?

R: Seu Sistema SQL mantém um registro de tudo que foi feito dentro de uma transação. É chamado de log de transações, ou registro de transações, e ele vai crescendo à medida que utiliza a transação. É melhor salvar com utilizando transações para quando realmente precisar estar apto a desfazer algo para evitar o gasto de espaço fazendo com que seu Sistema SQL tenha que trabalhar mais do que o necessário para ficar em dia daquilo que você já fez.



**Há alguma forma para que Greg tenha completo controle de quem faz o quê nas tabelas de seu banco de dados?**

**Vá para o próximo capítulo para descobrir.**



## Sua caixa de ferramentas SQL

**Você já tem o Capítulo 11 no bolso e já está quase enchendo sua caixa de ferramentas. Viu como utilizar o VIEW para seus dados e executar TRANSACTIONS (transações). Para uma lista completa de dicas neste livro, veja o Apêndice iii.**

### TRANSACTIONS (TRANSAÇÕES)

Este é um grupo de consultas que deve ser executado em conjunto, como uma unidade. Se elas não podem ser todo executadas sem interrupção, então nenhuma delas será executada.

START TRANSACTION é utilizado para dizer ao seu Sistema SQL para iniciar uma transação. Nada é permanente até se inserir um COMMIT. A transação irá continuar até ser confirmada ou ser utilizado o comando ROLLBACK para retroceder ao estado anterior do banco de dados antes do comando START TRANSACTION.

### VIEWS

Utilize uma view para tratar os resultados de uma consulta como uma tabela. Ótima para tornar consultas complexas em consultas simples.

### UPDATABLE VIEW (VIEWS ATUALIZÁVEIS)

Estas views permitem que você altere os dados das tabelas bases. Estas views devem conter todas as linhas com valores todos do tipo NOT NULL nas tabelas bases ou tabelas comuns.

### NON-UPDATABLE VIEWS (VIEWS NÃO-ATUALIZÁVEIS)

Views que não podem ser utilizadas com comandos de INSERT e UPDATE nas tabelas bases.

### CHECK CONSTRAINT

Utilize esta ferramenta apenas para determinar valores específicos a serem inseridos ou atualizados em uma tabela.

### CHECK OPTION

Utilize esta ferramenta quando criar uma view atualizável para forçar todos os dados inseridos e atualizações para satisfazerem uma cláusula WHERE dentro da view.

Aponte seu lápis

Solução  
da página 381.

Se ele executar o SELECT da página 381 utilizando a nova view aumento\_salario, como Frank poderá ordenar os resultados alfabeticamente pelo sobrenome?

*Adicione um ORDER BY sobrenome para ambas as views quando ela for criada ou SELECT quando for utilizar a view.*



## Aponte seu lápis

### Solução

da página 396.

Preencha as lacunas dos conteúdos após as transações.  
Aqui está como ela se encontra atualmente:

cofrinho

id	moeda	moeda_ano
1	V	1950
2	U	1972
3	C	2005
4	V	1999

START TRANSACTION;

```
UPDATE cofrinho set moeda = 'V' where moeda = 'U'  
AND moeda_ano < 1970;  
COMMIT;
```

*Sem combinação, portanto, sem alterações.*

id	moeda	moeda_ano
1	V	1950
2	U	1972
3	C	2005
4	V	1999

START TRANSACTION;

```
UPDATE cofrinho set moeda = 'C' where moeda = 'V';  
ROLLBACK;
```

*ROLLBACK, sem alterações.*

id	moeda	moeda_ano
1	V	1950
2	U	1972
3	C	2005
4	V	1999

START TRANSACTION;

```
UPDATE cofrinho set moeda = 'V' where moeda = 'C'  
AND moeda_ano > 1950;
```

*ROLLBACK, sem alterações.*

id	moeda	moeda_ano
1	U	1950
2	U	1972
3	C	2005
4	U	1999

START TRANSACTION;

```
UPDATE cofrinho set moeda = 'D' where moeda = 'V'  
AND moeda_ano > 1980;  
COMMIT;
```

*Esta linha é afetada.*

id	moeda	moeda_ano
1	V	1950
2	U	1972
3	C	2005
4	V	1999

START TRANSACTION;

```
UPDATE cofrinho set moeda = 'U' where moeda = 'C'  
AND moeda_ano > 1970;  
COMMIT;
```

*Esta linha é afetada.*

id	moeda	moeda_ano
1	V	1950
2	U	1972
3	U	2005
4	V	1999

## 12 Segurança

### Protegendo suas riquezas



**Você gastou uma enorme quantidade de tempo e energia para criar seu banco de dados.** E ficaria devastado se alguma coisa acontecesse com ela. Você também teve que dar acesso a seus dados para outras pessoas, e estava preocupado com o que eles pudessem inserir ou atualizar algo incorretamente, ou ainda pior, **deletar dados errados**. Você está prestes a aprender 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 quê com seus dados**.

## Problema de usuários

O Rastreador de Palhaços decolou de tal forma que o Conselho da Cidade de Dataville teve que雇用 um time completo de pessoas para rastrear palhaços e adicionarem os dados no banco de dados `rastreador_palhacos`.

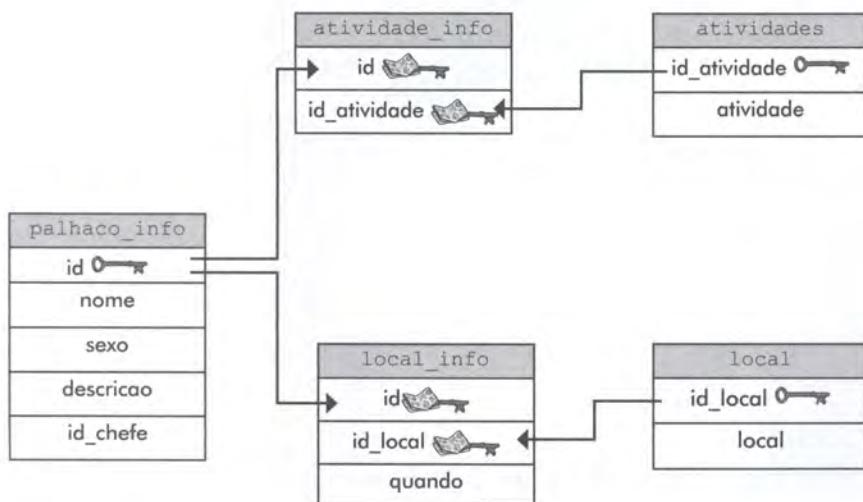
Infelizmente, o time foi infiltrado por um palhaço disfarçado em roupas comuns que utilizou o codinome “George”. Ele causou uma série de problemas no banco de dados, incluindo perda de dados, dados modificados e dados duplicados que só existem porque ele deliberadamente digitou de forma errada. Aqui estão alguns dos problemas com o banco de dados rastreador de palhaços:



**Snuggles, Snugles e Snuggels, todos têm linhas na tabela `palhaço_info`. Temos quase certeza que eles são na verdade o mesmo palhaço porque as colunas sexo e descrição são idênticas (exceto pelos erros de digitação).**

**Com estes múltiplos dados inseridos na tabela `palhaço_info`, temos uma bagunça na nossas informações acerca da visualização. A tabela `local_info` utiliza o ID da `palhaco_info` de Snuggles, Snugles e Snuggels.**

**A tabela atividade também está cheia de erros de digitação. Snuggles é um malabrista, Snugles é um mabalarista, e Snuggels é um malaborista.**



## Evitando erros no banco de dados rastreador de palhaços

George se demitiu antes que qualquer um pudesse notar que ele estava sabotando os dados, agora fomos deixando para recolher os pedaços. De agora em diante, quando contratarmos novas pessoas, precisamos dar a eles a habilidade de utilizar o `SELECT` a partir do banco de dados para que possam identificar os palhaços, mas queremos mantê-los longe de utilizarem o `INSERT` para inserir dados. Ou ainda de atualizar os dados com `UPDATE`. Ou qualquer outra coisa até termos tempo para fazer uma checagem completa nos bastidores.

Nós também vamos precisar ser cuidadosos; ao pedir para novos empregados deletar dados para consertar os erros de George, eles podem acabar deletando dados úteis junto com os ruins.

É hora de proteger o banco de dados rastreador de palhaços antes que outros palhaços como George o destruam completamente.

 Aponte seu lápis

Proteja o banco de dados rastreador de palhaços de possíveis sabotagens de palhaços. Em cada lado, escreva alguma consulta que novos empregados devem ou não devem estar aptos a realizar. Inclua o nome das tabelas quando possível.

Novos empregados devem estar aptos para:

*exemplo: SELECT from atividades;*

Novos empregados não devem estar aptos para:

*exemplo: DROPTABLE na palhaco\_info;*

 Aponte seu lápis

## Solução

Proteja o banco de dados rastreador de palhaços de possíveis sabotagens de palhaços. Em cada lado, escreva alguma consulta que novos empregados devem ou não devem estar aptos a realizar. Inclua o nome das tabelas quando possível.

Novos empregados devem estar aptos para:

*exemplo: SELECT from atividades*

Novos empregados não devem estar aptos para:

*exemplo: DROPTABLE na palhaco\_info;*

*SELECT from palhaco\_info,  
atividades\_info,  
atividades\_local\_info;*

*DROPTABLE na palhaco\_info, atividades\_info,  
atividades\_local\_info;*

*INSERT para palhaco\_info, atividades\_info,  
atividades\_local\_info;*

*UPDATE para palhaco\_info, atividades\_info,  
atividades\_local\_info;*

*ALTER na palhaco\_info, atividades\_info,  
atividades\_local\_info;*

*DELETE a palhaco\_info, atividades\_info,  
atividades\_local\_info;*

**Há boas notícias! Nós podemos impedir que palhaços como George destruam nossos dados!**

SQL nos dá a habilidade de controlar quais dos nossos empregados podem ou não podem fazer com o banco de dados rastreador de palhaços. Antes de estarmos aptos a fazer isso, precisamos dar a ele, e a cada outro que utilize nosso banco de dados uma **conta de usuário**.



## Proteja a conta de usuários raiz

Até aqui, tivemos apenas um usuário em nosso banco de dados e nenhuma senha. Qualquer um com acesso ao um terminal ou à interface gráfica de nosso banco de dados tem controle completo sobre o banco de dados.

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. Não queremos limitar o que o usuário raiz pode fazer, mas queremos sim, dar ao nosso usuário raiz uma senha. Em MySQL, o comando é simples:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('b4dc10wnZ');
```

O nome do usuário raiz é simplesmente 'root'.

'localhost' indica que este é o lugar onde o Sistema SQL está instalando e sendo executado.

Esta é a senha que escolhemos para nosso usuário raiz.

Outras técnicas dos Sistemas variam. Oracle utiliza:

```
alter user root identified by new-password;
```

Se estiver utilizando uma interface gráfica para seu banco de dados, você provavelmente encontrará um jeito mais fácil e direto para alterar as senhas. **O mais importante não é como fazê-lo, mas o que definitivamente deve fazê-lo.**

Consulte a documentação específica de seu Sistema SQL para informações sobre proteção da conta raiz.

---

não existem  
Perguntas Idiotas

---

**P:** Ainda não estou entendendo o que "localhost" quer dizer. Você pode explicar com mais detalhes?

**R:** localhost quer dizer que o computador que você está utilizando para executar suas consultas é o mesmo computador onde o Sistema SQL está instalado. localhost é o valor padrão para este parâmetro, então inclui-lo na sintaxe é opcional.

**P:** Mas se eu estiver utilizando uma máquina cliente em algum outro lugar?

**R:** Isto é conhecido como acesso remoto. Você terá que dizer a consulta onde o computador está. Poderá fazer com um endereço IP ou um hostname ao invés de um localhost. Por exemplo, se o seu Sistema SQL foi instalado em uma máquina chamada kumquats na rede O'Reilly, você poderá usar algo como root@kumquats.oreilly.com. Mas este não é um servidor de verdade, então é claro não irá funcionar.

## Crie um novo usuário

Há uma pergunta com uma resposta óbvia para você:

**Como você acha que o SQL armazena informações sobre usuários?**

Uma tabela, é claro! SQL mantém um banco de dados sobre ele mesmo. Isto inclui IDs de usuários, senhas e o que cada usuário está permitido a fazer em cada banco de dados.

Para criar um novo usuário, podemos começar com o nome de usuário e uma senha. Não há nenhum comando atual em SQL para criar usuário, mas a maioria dos Sistemas utilizará mais ou menos assim:

```
CREATE USER elsie
```

```
IDENTIFIED BY 'cl3v3rp4s5w0rd';
```

Aqui está o usuário para a mais nova funcionária, Elsie.

Aqui está a senha dela.



Você não poderia ter restringido Elsie de certas tabelas ao mesmo tempo em que criou a conta dela?



### Veja Isto!

**SQL não especifica como gerenciar os usuários.**

A criação de usuários varia de Sistema para Sistema. Você precisa checar sua documentação para encontrar a forma correta para criar um usuário no seu Sistema SQL.

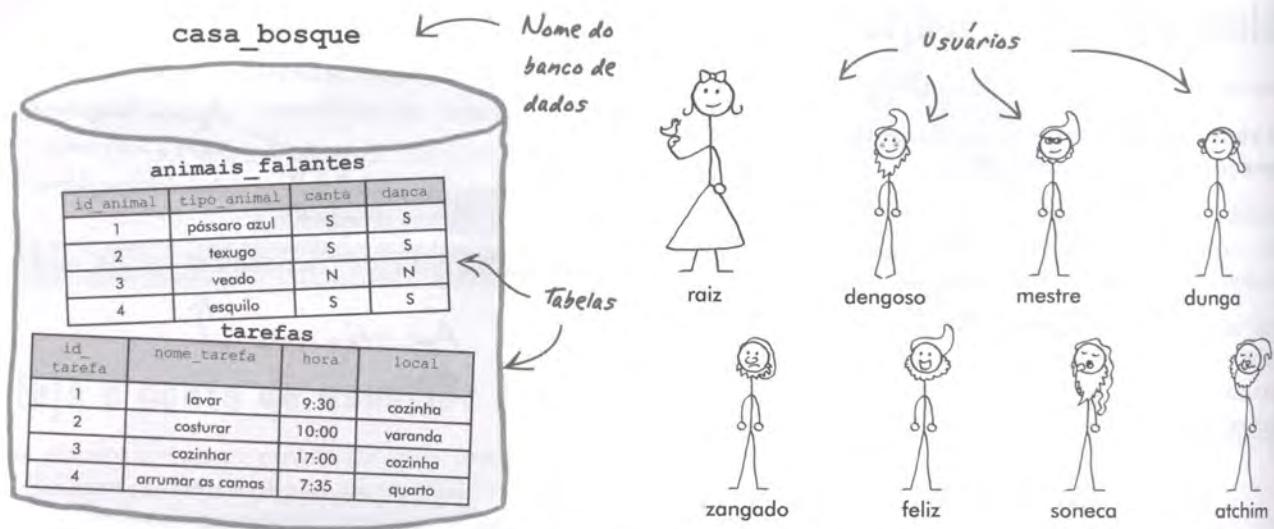
**Poderíamos, mas, às vezes, não sabemos que acessos precisamos permitir logo a princípio.**

Mas ainda precisamos decidir exatamente **ao que nosso usuário terá acesso**. Faremos uma coisa de cada vez. Vamos criar um usuário e depois permitir seu acesso especificamente ao qual ele precisa. Depois colocaremos tudo junto antes de terminarmos. A vantagem de saber como dar acesso independentemente de criar um usuário é o que nos dá a habilidade de fazer alterações a acessos de usuários posteriormente na medida que nosso banco de dados é alterado.

## Decida exatamente o que o usuário necessita

Criamos a conta de Elsie. Da forma que ela está neste momento, ela não tem permissão para nada. Temos que utilizar um comando **GRANT** (permissão) para dar permissão a ela até para o comando **SELECT** na tabela `palhaco_info`.

Como dono da nossa conta de raiz, que tem permissão para executar qualquer comando SQL, em qualquer coisa dentro do banco de dados, os novos usuários que criamos não tem qualquer permissão. O comando GRANT pode ser utilizado para dar direitos específicos para usuários de um banco de dados. Aqui está o que o comando GRANT nos permite fazer:



### Apenas usuários específicos podem alterar tabelas específicas.

Apenas a pessoa no comando poderia estar apta a adicionar novas tarefas à tabela tarefas. Apenas **raiz** pode utilizar o INSERT, UPDATE e DELETE nas tarefas. Entretanto, **feliz** está na chefia da tabela animais\_falantes e pode alterar a estrutura desta, bem como realizar qualquer operação sobre ela.

### Os dados em uma tabela específica podem ser acessados por apenas certos usuários.

Todos, exceto **zangado**, podem utilizar o SELECT a partir da tabela animais\_falantes. Ele não gosta de animais falantes.

### Mesmo dentro das tabelas pode haver necessidade de permissões: alguns usuários podem ver certas colunas, mas não outras.

Todos, com exceção de **Dunga** podem ver a coluna instruções na tabela tarefas (ela apenas o deixa mais confuso).

## Um simples comando GRANT

Sabemos que Elsie não ter permissão alguma até este momento. Ela pode ter acesso ao Sistema SQL ao utilizar seu nome de usuário e senha, mas é só isso. Ela precisa ser capaz de utilizar o SELECT na tabela palhaco\_info, então podemos dar permissão a ela para fazer isso. Nós precisamos GRANT (conceder) permissão para Elsie. Utilizaremos este comando:

Foi dada a permissão ao usuário para SELECT...

```
GRANT SELECT ON
palhaco_info
TO elsie;
```

...na tabela que indicamos aqui.

E o nome de usuário ao qual estamos concedendo permissão é elsie.

também precisa de permissão para utilizar o SELECT nas tabelas rastreador de palhaços para que consiga utilizar joins e subconsultas em seus comandos SELECT. Precisamos de comando GRANT separado para cada tabela:

```
GRANT SELECT ON atividades TO elsie;
GRANT SELECT ON local TO elsie;
GRANT SELECT ON atividades_info TO elsie;
GRANT SELECT ON local_info TO elsie;
```



Exercícios

## O código

## O que o código faz?

1. GRANT INSERT ON animais\_magicos TO doc; .....  
.....
2. GRANT DELETE ON tarefas  
TO feliz, soneca; .....  
.....
3. GRANT DELETE ON tarefas  
TO feliz, soneca WITH  
GRANT OPTION; .....  
.....
4. GRANT SELECT (nome\_tarefa)  
ON tarefa TO dunga;  
*Dica: este é o nome de uma coluna.* .....  
.....
5. GRANT SELECT, INSERT  
ON animais\_falantes TO  
atchim; .....  
.....
6. GRANT ALL ON animais\_falantes  
TO dengoso; .....  
.....

Agora tente escrever alguns dos seus próprios códigos GRANT.

7. .....  
.....
8. .....  
.....
9. .....  
.....
10. .....  
.....

Dá permissão ao Doc para utilizar o SELECT em tarefas.

Dá permissão ao Soneca para utilizar o DELETE de animais\_falantes, e dá permissão ao Soneca para conceder permissão ao DELETE de animais\_falantes para qualquer outra pessoa.

Dá a TODOS os usuários todas as permissões para tarefas.

Este comando permite que defina privilégio para o comando SELECT para Doc de uma só vez para todas as tabelas no banco de dados casa\_bosque.



## Solução dos Exercícios

### O código

1. GRANT INSERT ON animais\_magicos TO doc;
2. GRANT DELETE ON tarefas TO feliz, soneca;
3. GRANT DELETE ON tarefas TO feliz, soneca WITH GRANT OPTION;
4. GRANT SELECT (nome\_tarefa) ON tarefa TO dunga;
5. GRANT SELECT, INSERT ON animais\_falantes TO atchim;
6. GRANT ALL ON animais\_falantes TO dengoso;

### O que o código faz?

Permite a Doc utilizar o INSERT na tabela animais\_magicos.

Permite ao feliz e ao soneca utilizarem o DELETE na tabela tarefas.

Permite ao feliz e ao soneca utilizarem o DELETE na tabela tarefas e também para dar a outros a mesma permissão.

Permite ao dunga utilizar o SELECT apenas na coluna nome\_tarefa da tabela tarefas.

Permite ao Atchim utilizar SELECT e INSERT na tabela animais\_falantes.

Permite ao dengoso utilizar o SELECT, UPDATE, INSERT e DELETE na tabela animais\_falantes.

Agora tente escrever alguns dos seus próprios códigos GRANT.

7. GRANT SELECT ON tarefas TO doc;
8. GRANT DELETE ON animais\_falantes TO soneca WITH GRANT OPTION;
9. GRANT ALL ON tarefas TO dengoso, mestre, dunga, zangado, feliz, soneca, atchim;
10. GRANT SELECT ON casa\_bosque.\* TO doc

Dá permissão ao Doc para utilizar o SELECT em tarefas.

Dá permissão ao Soneca para utilizar o DELETE de animais\_falantes, e dá permissão ao Soneca para conceder permissão ao DELETE de animais\_falantes para qualquer outra pessoa.

Dá a TODOS os usuários todas as permissões para tarefas.

Este comando permite que defina privilégio para o comando SELECT para Doc de uma só vez para todas as tabelas no banco de dados casa\_bosque.

## Funções do comando GRANT

No exercício que acabou de fazer, você viu as principais funções do comando GRANT. Aqui estão elas:

### 1 Você pode nomear múltiplos usuários no mesmo comando GRANT.

Cada um dos usuários nomeados obterá a mesma permissão concedida a eles.

### 2 WITH GRANT OPTION dá permissão aos usuários para dar a outros usuários a permissão que a eles foi concedida.

Parece confuso, mas simplesmente quer dizer que se o usuário tem permissão de utilizar o SELECT na tabela tarefas, ele pode dar esta permissão a qualquer outro usuário para utilizar o SELECT na tabela tarefas.

### 3 Uma coluna, ou colunas específicas em uma tabela, pode ser utilizada ao invés da tabela inteira.

A permissão pode ser dada para utilizar o SELECT apenas em uma só coluna. O único resultado que o usuário verá será aquela coluna.

### 4 Você pode especificar mais de uma permissão para uma tabela.

Como qualquer outra permissão que você queira conceder em uma tabela, utilizando vírgula para cada uma delas.

### 5 GRANT ALL dá permissão ao usuário para utilizar o SELECT, UPDATE, INSERT e DELETE daquela tabela específica.

É simplesmente uma forma abreviada de dizer “dê aos usuários permissão para utilizar o SELECT, UPDATE, INSERT e DELETE daquela tabela específica”.

### 6 Você pode especificar a todas as tabelas em um banco de dados com o nome\_bancodedados \*.

Da mesma forma que você utiliza o \* coringa em um comando SELECT, este especifica todas as tabelas em um banco de dados.

# REVOKE (REVOGUE) privilégios

Suponha que decidimos remover o privilégio de utilizar o SELECT dado a Elsie. Para fazer isso, precisamos do comando REVOKE.

Ainda se lembra do nosso simples comando GRANT? A sintaxe para o REVOKE é quase idêntica. Ao invés de utilizar a palavra “grant”, utiliza-se “revoke”, e ao invés de utilizar “to”, utiliza-se “from”.

Estamos removendo o privilégio para o SELECT.

```
REVOKE SELECT ON
palhaco_info
FROM elsie;
```

Revogamos o privilégio de um usuário ao invés de concedê-lo.

Você pode revogar parte do privilégio como “WITH GRANT OPTION” **mas deixar o privilégio intacto**. Neste exemplo, **feliz** e **soneca** ainda poderão utilizar o DELETE na tabela tarefas, mas não poderão dar a mais ninguém o mesmo privilégio:

Estamos revogando apenas o privilégio GRANT OPTION.

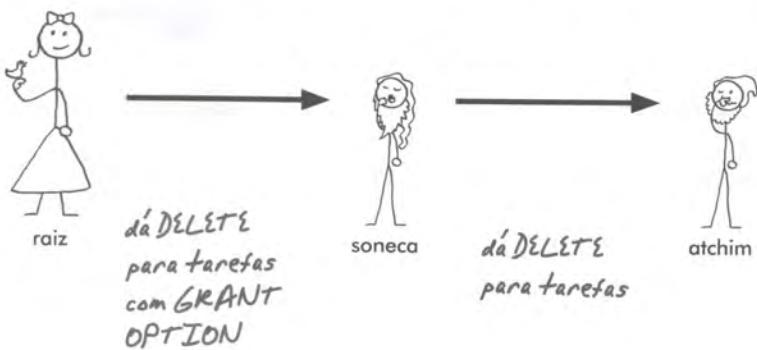
```
REVOKE GRANT OPTION ON
DELETE ON tarefas
FROM feliz, soneca;
```

Os usuários **feliz** e **soneca** ainda poderão utilizar o **DELETE**, mas não poderão conceder a ninguém mais este privilégio.

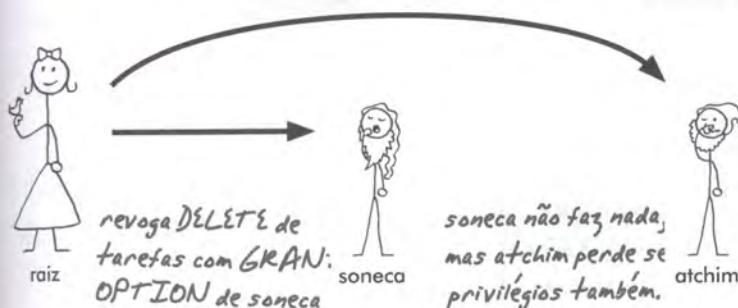


## REVOGANDO uma GRANT OPTION usada

Considere o seguinte cenário. O usuário raiz deu ao soneca privilégios do DELETE com GRANT OPTION na tabela tarefas. Depois **soneca** deu a **atchim** estes mesmos privilégios também.



...nha que o usuário raiz mude de idéia e tire o privilégio de **soneca**. Ele também  
removido de **atchim**, muito embora ela tenha revogado de **soneca**.



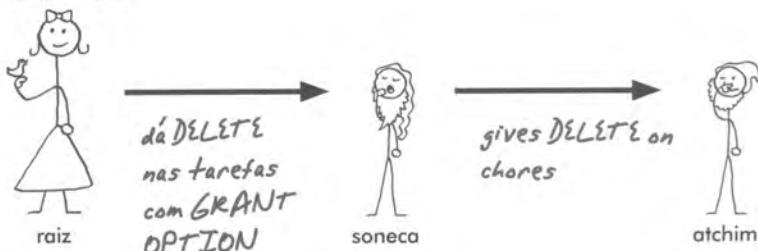
O efeito colateral do comando REVOKE foi que  
também perdeu seu privilégio. Há duas palavras-  
que você pode usar que permitirá que controle o  
que aconteça quando estiver revogando.

### PODER DO CÉREBRO

Você está prestes a conhecer as palavras-chaves RESTRICT e CASCADE. O que acha que cada uma faz?

## REVOGANDO com mais precisão

duas maneiras de revogar privilégios e  
se que não estará afetando mais  
do que queira. Você pode utilizar  
avras-chave CASCADE e RESTRICT  
a direcionar quem mantém e quem  
os privilégios de forma mais precisa.



primeiro, CASCADE, remove o privilégio do usuário que você indicou (neste caso,  
da mesma forma que ninguém mais além daquele usuário deu permissão).

**REVOKE DELETE ON tarefas FROM soneca CASCADE;**



Utilizando RESTRICT quando quiser remover um privilégio de um usuário que retornará um erro se o usuário tiver concedido privilégio a alguém.

```
REVOKE DELETE ON tarefas FROM soneca RESTRICT;
```



Ambos retêm privilégios e raiz recebe um erro. Ela é impedida de fazer a alteração e obtém um erro porque isto também terá um efeito em **atchim**.



Aponte seu lápis

Alguém fica constantemente dando privilégios errados para Elsie. Escreva os comandos REVOKE apropriados para retornar a ela o status para status seguro de ter apenas o privilégio ao SELECT.

```
GRANT SELECT, INSERT, DELETE ON local TO elsie;
```

```
.....
```

```
GRANT ALL ON palhaco_info TO elsie;
```

```
.....
```

```
GRANT SELECT, INSERT ON atividades TO elsie;
```

```
GRANT DELETE, SELECT ON info_local TO elsie  
WITH GRANT OPTION;
```

```
GRANT INSERT(local), DELETE ON local TO elsie;
```

## Aponte seu lápis

### Solução

Alguém fica constantemente dando privilégios errados para Elsie. Escreva os comandos REVOKE apropriados para retornar a ela o status para status seguro de ter apenas o privilégio ao SELECT.

GRANT SELECT, INSERT, DELETE ON local TO elsie;

~~REVOKE INSERT, UPDATE, DELETE ON local FROM elsie;~~

GRANT ALL ON palhaco\_info TO elsie;

~~REVOKE INSERT, UPDATE, DELETE ON palhaco\_info FROM elsie;~~

GRANT SELECT, INSERT ON atividades TO elsie;

~~REVOKE INSERT ON atividades FROM elsie;~~

GRANT DELETE, SELECT ON info\_local TO elsie  
WITH GRANT OPTION;

~~REVOKE INSERT ON atividades FROM elsie CASCADE;~~

GRANT INSERT(local), DELETE ON local TO elsie;

~~REVOKE GRANT INSERT(local) DELETE ON local FROM  
elsie;~~

Queremos deixá-la apenas com o privilégio para o SELECT, para que não tenha seus privilégios revogados tudo novamente.

Outra forma que você poderia ter feito isso era REVOKE tudo e então conceder privilégio do que precisasse.

Parece que poderíamos utilizar o GRANT aqui também para nos certificar que ela ainda pode utilizar o SELECT para selecionar os locais.

E seria melhor certificarmos se ela não deu privilégio das quais ela tinha a mais nenhuma.

## não existem Perguntas Idiotas

Ainda estou pensando nos comandos GRANT que especificam os nomes das colunas. O que acontece se você conceder privilégios para inserir o INSERT em apenas uma coluna da sua?

Boa pergunta. Este é na verdade um privilégio útil a se ter. Se você pode colocar valores em uma coluna, não pode inserir uma linha completa, de fato. A única forma disto funcionar é se a tabela tiver apenas uma coluna e ela estiver especificada no comando GRANT.

Existem quaisquer outros comandos que sejam úteis quanto?

Quase todos os privilégios concedidos a uma só coluna são bastante inúteis a não ser que estejam em conjunto com um comando SELECT em uma GRANT.

P: Suponha que eu queira adicionar um usuário e permitir que ele utilize o SELECT de todas as tabelas em todo o meu banco de dados. Existe alguma forma fácil de fazer isso?

R: Como muitas outras coisas neste capítulo, ele depende do Sistema SQL. Você pode conceder benefícios globais no MySQL da seguinte forma:

```
GRANT SELECT ON *.*  
TO elsie;
```

P: Então a palavra-chave CASCADE é o padrão no caso de não especificar como você quer utilizar o REVOKE?

R: Geralmente, CASCADE é o padrão, mas mais uma vez, cheque no seu sistema para respostas mais específicas.

P: O que acontece se eu utilizar o REVOKE para revogar algo com o qual o usuário não devia ter iniciado?

R: Você vai simplesmente obter um erro dizendo que o GRANT não existe em primeiro lugar.

P: O que acontece se duas pessoas diferentes concederem ao usuário atchim o mesmo privilégio que o usuário raiz está revogando no exemplo anterior?

R: É aí que as coisas começam a ficar complicadas. Alguns sistemas não irão prestar atenção de onde o GRANT está vindo quando o modo CASCADE é usado, e alguns vão ainda ignorar este fato. Este é mais um caso de se checar a documentação específica do software que você utiliza.

P: Há algo mais, além de tabelas e colunas onde eu possa utilizar o GRANT e o REVOKE?

R: Você pode utilizar estes comandos com uma view da mesma forma que o faria com uma tabela, exceto se a view for do tipo não atualizável. Neste caso, você não estaria apto a utilizar o INSERT ainda que tivesse permissão para isso. E da mesma forma que uma tabela, você pode conceder acesso para colunas específicas da view.



Então se eu quiser cinco usuários diferentes tendo as mesmas permissões, só os adiciono através de vírgulas ao final do comando GRANT, certo?

### Isto vai funcionar. E quando você tem uma pequena quantidade de usuários, esta é a forma de se fazer.

Mas à medida que a organização cresce, você começará a ter classes de usuários. Poderá ter 10 usuários exclusivos para inserirem os dados, e precisam apenas inserir e selecionar algumas tabelas. Você poderá ter também três usuários com poder de fazer qualquer coisa, e muitos usuários que precisam apenas utilizar o SELECT. Ainda poderá ter um software e aplicações web que contatem seu banco de dados e precisem consultar views específicas de formas específicas.



Então se você pode criar classes, porque não criar um usuário só para cada classe de pessoas acima e permitir que elas compartilhem o mesmo nome de usuário e senha?

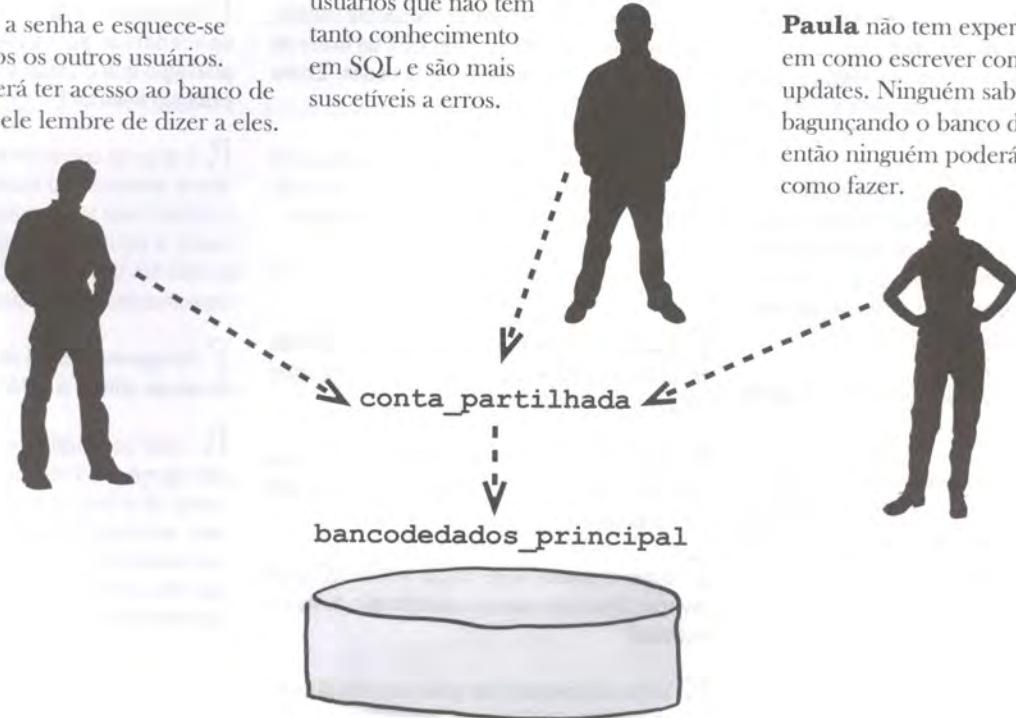
## O problema com contas partilhadas

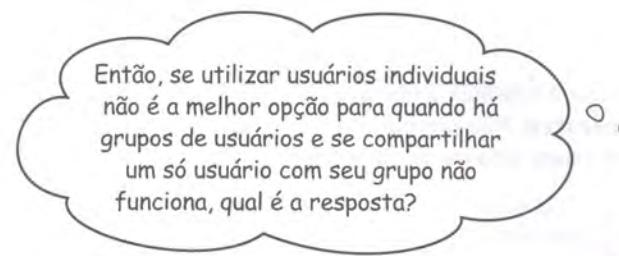
Enquanto algumas companhias se dão muito bem com a existência de um só usuário que possa acessar banco de dados, esta não é a forma mais segura de se fazer as coisas. Aqui está um exemplo do que poderia ir errado.

**Randy** tem que completar os privilégios para tudo no banco de dados para poder fazer seu trabalho. Isto faz com que o banco de dados fique vulnerável para outros usuários que não têm tanto conhecimento em SQL e são mais suscetíveis a erros.

**Simon** altera a senha e esquece-se de dizer a todos os outros usuários. Ninguém poderá ter acesso ao banco de dados até que ele lembre de dizer a eles.

**Paula** não tem experiência em como escrever comandos updates. Ninguém sabe quem está bagunçando o banco de dados, então ninguém poderá ensiná-la como fazer.





**Você precisa de uma forma de dar aos grupos os privilégios que eles precisam, enquanto ao mesmo tempo atribui uma conta individual para cada usuário.**

Do que você precisa são **roles (funções)**. Uma role é uma forma de agrupar privilégios específicos e aplicá-los a todos em um grupo. Sua role se torna um objeto no banco de dados que pode alterar conforme necessário quando seu banco de dados muda sem ter que alterar explicitamente os privilégios de cada usuário, individualmente para refletir as alterações ocorridas pelo banco de dados.

Criar uma role é muito fácil:

```
CREATE ROLE entrada_dados;
```

O nome da role que você está criando.

Para adicionar privilégios a uma role, trata-a como se fosse simplesmente um usuário:



### Veja Isto!

**Não existem roles no MySQL**  
Roles são características que uma futura versão do MySQL provavelmente terá, mas por ora, você terá que definir suas tarefas no sistema individual por usuário.

```
GRANT SELECT, INSERT ON alguma_tabela TO entrada_dados;
```

**Nós já criamos nossa role e demos privilégios a você. Agora precisamos atribuir a role a alguém...**

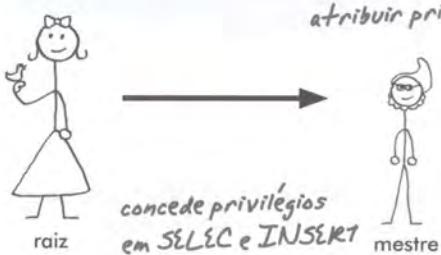
Ao invés do usuário, utilizamos o nome da role quando queremos atribuir privilégios.

## Utilizando sua role

Se criarmos nossa role, poderíamos ter dado privilégios a nossos usuários que inserem dados diretamente, utilizando comando GRANT da seguinte forma:

```
GRANT SELECT, INSERT  
ON animais_falantes  
TO mestre;
```

A forma antiga.

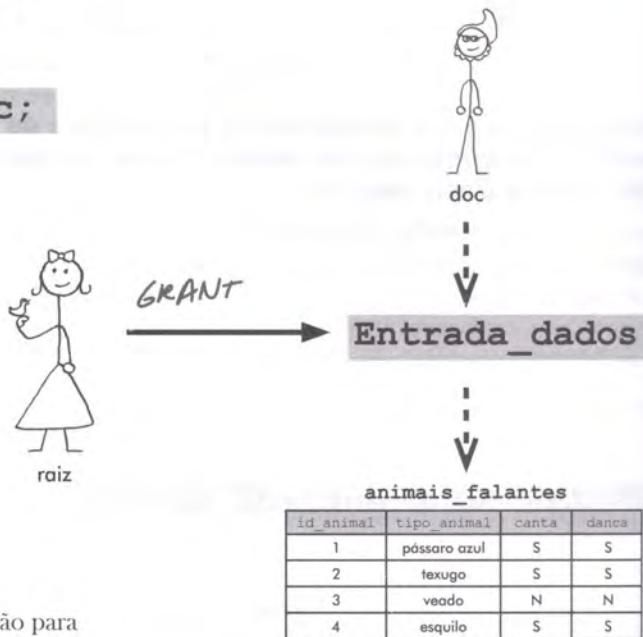


id_animal	tipo_animal	canta	danca
1	pássaro azul	S	S
2	texugo	S	S
3	veado	N	N
4	esquilo	S	S

Agora tudo o que precisamos fazer é substituir a operação GRANT por nossa nova role e aplicá-la ao **mestre**. Não precisamos mencionar os privilégios ou tabelas porque estará tudo armazenado na role `entrada_dados`:

```
GRANT entrada_dados TO doc;
```

O nome da role substitui o nome da tabela e os privilégios.



## Eliminando roles

Quando você não mais precisar de sua role, não há mais razão para mantê-la. Use um comando DROP para se ver livre dela.

```
DROP ROLE entrada_dados;
```

---

### não existem Perguntas Idiotas

---

**P:** E se eu quiser conceder privilégios para todas as tabelas em seu banco de dados? Preciso digitar cada um deles?

**R:** Não, você pode digitar esta sintaxe:

```
GRANT SELECT, INSERT, DELETE
ON gregs_list.*
TO jim;
```

Apenas cite o nome do banco de dados e utilize \* para atribuir privilégios para todas as tabelas no banco de dados.

**P:** Se uma role está atribuída para um usuário, ainda assim, posso deletá-la?

**R:** Você pode eliminar roles que estiverem em uso. Tenha muito cuidado ao deletar uma role e assim corte os privilégios de usuários que precisem deles.

**P:** Quer dizer que se um usuário tiver uma role que for deletada, ele perde suas permissões

**R:** É exatamente isso. É como se tivesse explicitamente concedido os privilégios e depois os tivesse revogado. Entretanto, ao invés de afetar apenas um usuário quando você revoga seus privilégios, você terá afetado as permissões de todos os usuários vinculadas a esta role.

**P:** Um usuário pode ter mais que uma role ao mesmo tempo?

**R:** Sim, apenas certifique-se que não tenha permissões conflitantes ou poderá causar alguns problemas. As permissões negadas prevalecem sobre aquelas que são concedidas.

Aponte seu lápis

**Revogando sua role**

Revogar sua role funciona de forma semelhante a revogar uma Grant. Veja se você consegue escrever o comando para revogar entrada\_dados de Mestre **sem olhar de volta no capítulo.**

Aponte seu lápis

Solução

Revogar sua role funciona de forma semelhante a revogar uma Grant. Veja se você consegue escrever o comando para revogar entrada\_dados de Mestre **sem olhar de volta no capítulo.**

REVOKE entrada\_dados FROM mestre;

## Utilizando sua role WITH ADMIN OPTION (com opção para administrador)

Nosso comando GRANT possui WITH GRANT OPTION, a role tem o comando similar WITH ADMIN OPTION. Esta opção permite a qualquer um com aquela role concedê-la a qualquer outra pessoa. Por exemplo, se você utilizar este comando:

GRANT entrada\_dados TO mestre WITH ADMIN OPTION;

Mestre agora tem privilégios de administrador, e podemos conceder a role entrada\_dados da mesma forma que ela foi concedida a ele.

GRANT entrada\_dados TO feliz;

WITH ADMIN OPTION permite ao usuário mestre conceder a role entrada\_dados para qualquer outro usuário.

Como utilizada uma role, o comando REVOKE possui as mesmas opções CASCADE e RESTRICT. Vamos dar uma olhada como elas funcionam:

## Revogue a role no modo CASCADE

Usado com a CASCADE, o comando REVOKE afeta qualquer um na corrente, da mesma forma que afeta o alvo original:

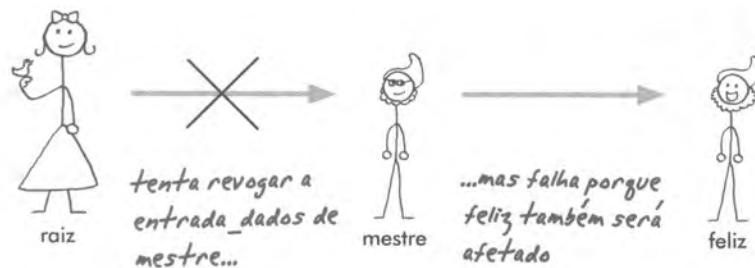
**REVOKE entrada\_dados FROM mestre CASCADE;**



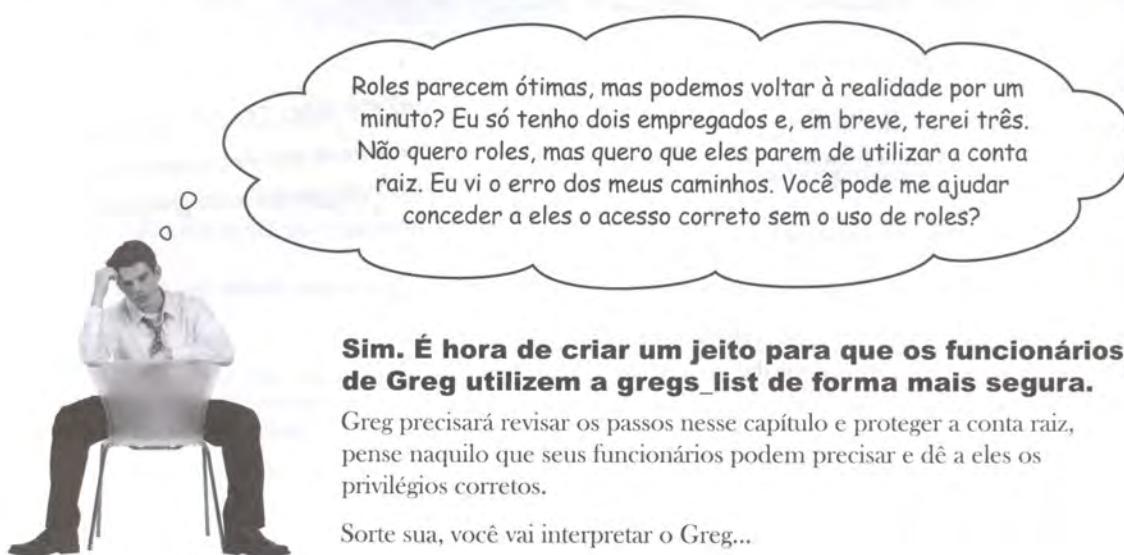
## Revogue a role com RESTRICT

Utilizando a palavra-chave RESTRICT quando você quer remover o privilégio de um usuário irá retornar um erro se o usuário tiver concedido privilégios para alguma outra pessoa.

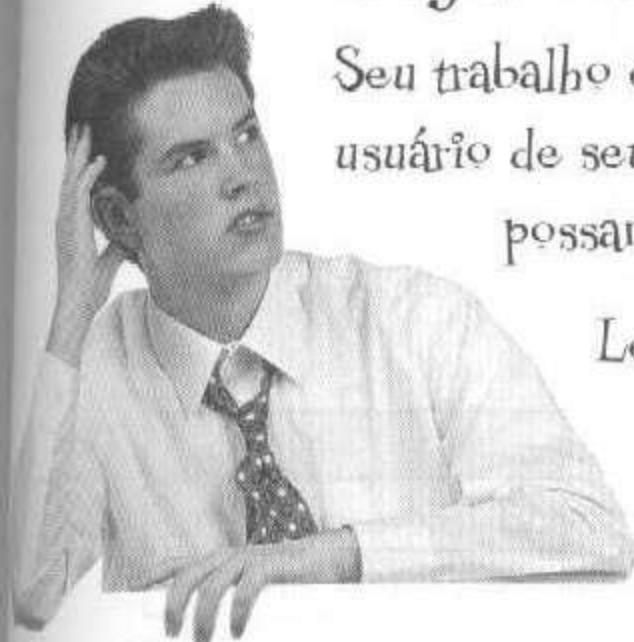
**REVOKE entrada\_dados FROM mestre RESTRICT;**



Ambos retêm os privilégios e raiz recebe um erro. Ela é impedida de fazer a mudança porque isto terá efeito em feliz também.



## Seja Greg



Seu trabalho é interpretar Greg pela última vez e consertar o lado usuário de seu banco de dados para que seus empregados não possam bagunçar qualquer coisa acidentalmente.

Leia as descrições dos trabalhos para cada usuário e invente com múltiplos comandos `GRANT` que deem a eles os dados que precisam enquanto não se deixe acessarem qualquer coisa que não poderiam.



**Frank:** "Eu sou responsável por encontrar empregos compatíveis com futuras vagas de emprego. Nunca inseri nada no banco de dados, embora eu delete listas de empregos quando encontro combinações ou quando a vaga é preenchida. Preciso olhar informações sobre os contatos na `meus_contatos`, às vezes".

**Jim:** "Eu insiro todos os novos dados em todo o banco de dados. Fiquei muito bom em inserir dados, agora que não posso inserir acidentalmente X para sexo. Eu também atualizo os dados. Estou aprendendo a deletar, mas até agora Greg tem dito para eu não fazer isto. É claro, o que ele não sabe..."

**Joe:** "Eu acabo de ser contratado por Greg para gerenciar a parte comercial das coisas. Ele quer integrar suas informações do contato em uma web site. Sou mais um programador em Web que um garoto SQL, mas posso criar selects simples. Eu não insiro dados E nem Windows. Desculpe, piada de mau gosto".

De uma olhada no banco de dados `gregs_list` e de a estes garotos algumas concessões antes que eles estraguem alguns dados.

Escreva o comando para dar ao usuário conhecido como "root" uma senha.

.....

.....

.....

Escreva três comandos para criar contas de usuários para cada um dos três empregados.

.....

.....

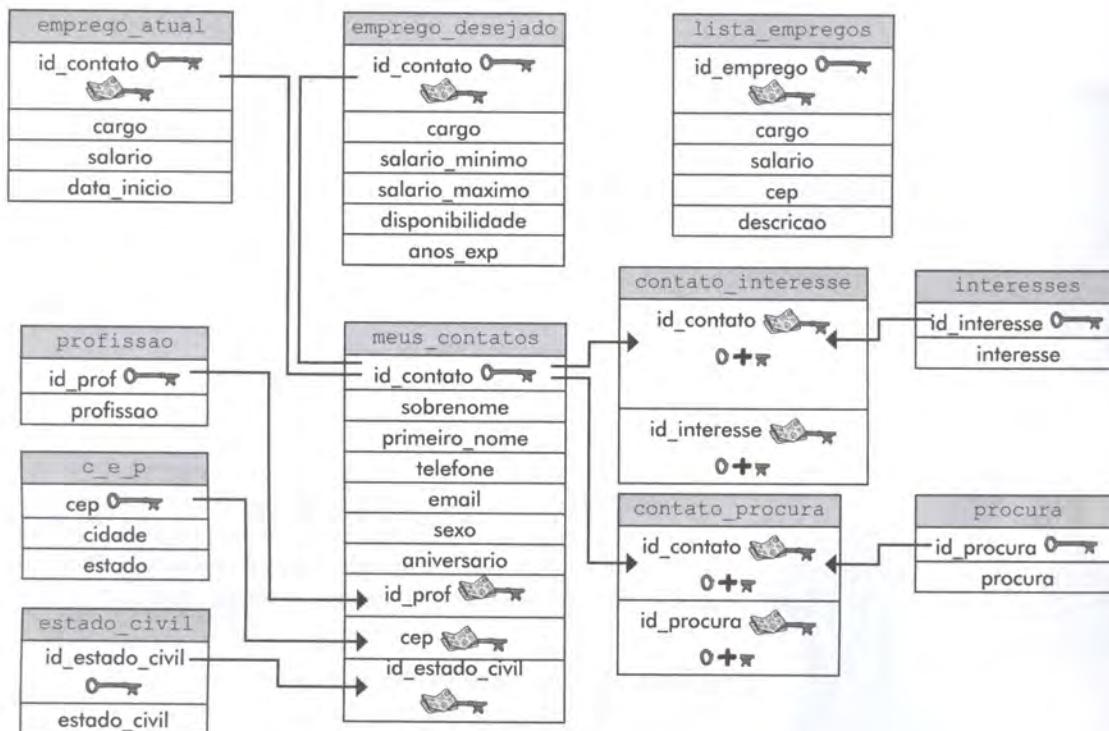
.....

Escreva comandos `GRANT` para cada novo empregado para dar a eles as permissões corretas.

.....

.....

.....



## Seja Greg



Seu trabalho é interpretar Greg pela última vez e consertar o lado usuário de seu banco de dados para que seus empregados não possam bagunçar qualquer coisa acidentalmente.

Leia as descrições dos trabalhos para cada usuário e invente com múltiplos comandos GRANT que deem a eles os dados que precisam enquanto não se deixa acessarem qualquer coisa que não poderiam.

Escreva o comando para dar ao usuário conhecido como "root" uma senha.

```
SETPASSWORD FOR root@localhost = PASSWORD('gr3GReuz');
```

Escreva três comandos para criar contas de usuários para cada um dos três empregados.

```
CREATE USER frank IDENTIFIED BY 'j0bM4tch';
```

```
CREATE USER jim IDENTIFIED BY 'N0m0r3xs';
```

```
CREATE USER joe IDENTIFIED BY 's3lect100d';
```

Não se preocupe se suas senhas são diferentes, contanto que você tenha juntado as partes deste comando na ordem correta, você está pronto!

Escreva comandos GRANT para cada novo empregado para dar a eles as permissões corretas.

```
GRANT DELETE ON lista_empregos TO frank;
```

```
GRANT SELECT ON meus_contatos * TO frank;
```

```
GRANT SELECT, INSERT ON gregs_list * TO jim;
```

```
GRANT SELECT ON meus_contatos, profissao, c_e_p,
estado_civil, contato_interesse, interesses, contato_procura, procura TO joe;
```

Frank precisa estar apto a remover listas de empregos e consultas (select) na tabela meus\_contatos.

Jim precisa de acesso ao SELECT e INSERT por toda a extensão do banco de dados gregs\_list. Por ora, o manteremos longe do DELETE.

Enquanto isso, Joe precisa estar apto a selecionar a partir das tabelas originais, mas não as tabelas que lidam com empregos.

## Combinação de CREATE USER e GRANT



Antes de você ir,  
tentaremos utilizar o comando  
CREATE USER e GRANT  
listados em um só comando?

**Sim, nós podemos. Tudo o que precisamos  
é combinar as duas partes que você já viu.**

Elas são o comando CREATE USER e GRANT que  
precisamos para Elsie.

```
CREATE USER elsie  
IDENTIFIED BY 'cl3v3rp4s5w0rd';
```

```
GRANT SELECT ON  
palhaco_info  
TO elsie;
```

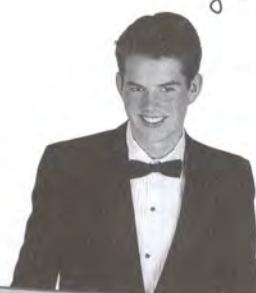
Podemos combinar os códigos e omitir a parte CREATE USER,  
porque a usuária elsie precisa ser criada antes que possa ter  
qualquer privilégio concedido a ela, seu Sistema SQL checar se  
ela existe, e se não, ele cria a conta dela automaticamente.

```
GRANT SELECT ON  
palhaco_info  
TO elsie;  
IDENTIFIED BY 'cl3v3rp4s5w0rd';
```

## A lista de Greg agora é global!

Graças a sua ajuda, Greg agora está tão confortável no uso do SQL - e ensinando Jim, Frank e Joe como usá-lo - que sua lista se expandiu para incluir classificados locais e fóruns de discussão da mesma forma.

E a melhor notícia de todas? Ela tem feito tanto sucesso em Dataville que mais de 500 cidades ao redor do mundo têm suas próprias listas, e Greg é notícia de capa!



Obrigado,  
companheiros, eu não teria  
conseguido sem vocês! Ei, tenho  
uma franquia aberta disponível na  
sua cidade... Vamos falar sobre a  
lista de Greg!

# O CONSULTOR SEMANAL

## A decolagem da Greg's List

### Franquias e Fóruns

Amigos e parentes disseram que a fama não mudou Greg nem um pouco.

**Por Troy Armstrong**  
Equipe de Escritores de O Consultor

**DATAVILLE** - O empresário local Greg chegou ao topo. Sua rede de banco de dados cresceu a partir de adesivos, para uma simples tabela, para um banco de dados multi-tabela que oferece procura de um par, empregos, e muito mais.

Se você gostaria de se juntar a esta diversão, visite:

[www.gregs-list.net](http://www.gregs-list.net)

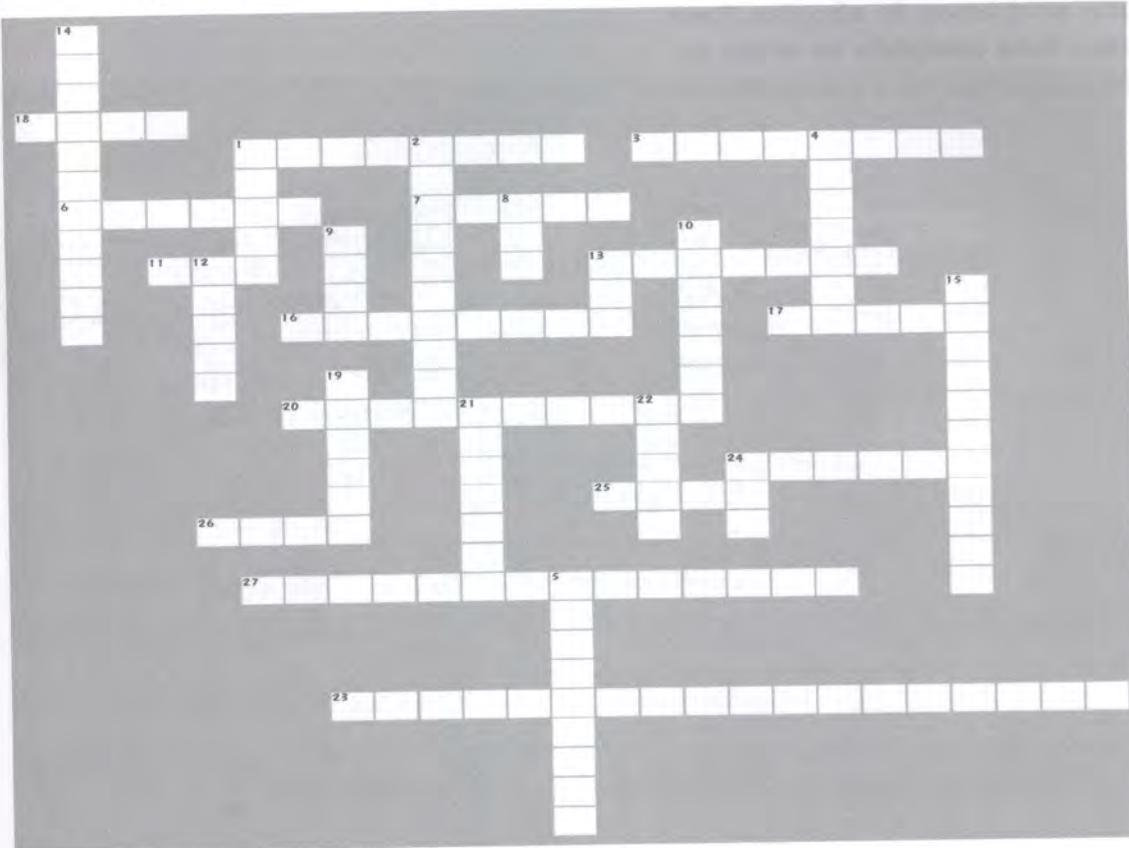


A lista Greg já chegou a sua cidade?  
É só uma questão de tempo, diz o  
analista de dados da cidade.



## (A última) Cruzada do SQL

Sim, é um dia triste, você está olhando para a última palavra-cruzada deste livro. Respire fundo...recheamos este com várias palavras-chave e comandos para fazer ele durar mais. Aproveite!



### Horizontais

- \_\_\_\_ dá ao usuário permissão para utilizar o SELECT, UPDATE, INSERT e DELETE de uma tabela específica.
- Esta função retorna cada valor único apenas uma vez, sem duplicidades.
- Tabelas \_\_\_\_ não deixarão dados duplicados, o que reduzirá o tamanho de seu banco de dados.
- Conceder uma role WITH \_\_\_\_ OPTION permite a um usuário a conceder uma role para qualquer outra pessoa.
- PASSWORD FOR = 'root'@'localhost' = PASSWORD('b4dcl0wnz');
- Valores armazenados em colunas CHAR ou VARCHAR são conhecidos desta forma.
- Utilizando \_\_\_\_ quando você quer remover um privilégio de um usuário retornará um erro se aquele usuário tiver concedido privilégios a mais alguém.
- Com uma conexão interna (inner join), você está comparando linhas de duas tabelas, mas a \_\_\_\_ daquelas duas tabelas não importa.
- Nós podemos utilizar uma \_\_\_\_ conexão para simular ter duas tabelas.
- Se alterar quaisquer das colunas não-chave pode causar a alteração de qualquer outra coluna, você tem uma \_\_\_\_ transitiva.
- Se a subconsulta funciona sozinha e não faz referência a consulta externa, ela é uma subconsulta \_\_\_\_.
- Isto quer dizer que seus dados foram partidos em pedaços menores de dados que não podem ou não devem ser divididos.
- Para ajudar você a se decidir quais passos em sua SQL podem ser considerados uma transação, lembre-se do acrônimo \_\_\_\_.
- Uma CONEXÃO EXTERNA (OUTER JOIN) pega todas as linhas na tabela esquerda e combina com as linhas da tabela DIREITA.
- Uma subconsulta \_\_\_\_ quer dizer que a consulta interna depende da consulta externa antes de poder ser resolvida.

### Verticais

- Você pode controlar exatamente o que os usuários podem fazer com tabelas e colunas com o comando \_\_\_\_.
- Uma dependência funcional \_\_\_\_ quer dizer que uma coluna não-chave é relacionada com quaisquer outras colunas não-chave.
- Você pode ter somente um campo AUTO INCREMENT por tabela, ela deve ser do tipo de dados \_\_\_\_.
- Uma CHAVE \_\_\_\_ é uma CHAVE PRIMÁRIA composta de múltiplas colunas, criando uma chave única.
- Você pode achar o maior valor na coluna com esta função.
- Atribuir isto é uma maneira que você tem de agrupar privilégios específicos, e aplicá-los a cada um daquele grupo.
- Use estas duas palavras para ordenar seus resultados alfabeticamente baseados na coluna que você especificar.
- A não-equijoin retorna quaisquer linhas que não sejam \_\_\_\_.
- Utilize esta cláusula em seu comando update para alterar um valor.
- Uma chave externa auto-\_\_\_\_ é uma chave primária de uma tabela utilizada naquela mesma tabela para outro propósito.
- Durante uma \_\_\_\_ , se todos os passos não forem completados sem interferência, nenhum deles deverá ser completados.
- Uma subconsulta é sempre um só comando \_\_\_\_.
- Estas conexões só funcionam se a coluna a qual você está conectando tem o mesmo nome em ambas as tabelas.
- Uma \_\_\_\_ constraint restringe quaisquer valores que você insira em uma coluna.
- Nossa tabela pode receber novas colunas com o comando ALTER e uma cláusula \_\_\_\_ COLUMN.



## Sua caixa de ferramentas SQL

Parabéns, você completou o Capítulo 12! Aprovei e reveja os princípios de segurança que acabamos de abordar. Para uma lista completa de dicas de ferramentas, veja o Apêndice iii.

**CREATE USER**  
utilizado por alguns Sistemas SQL para permitir que você crie um usuário e atribua uma senha.

**REVOKE**  
utilize este comando para revogar privilégios de um usuário.

**WITH GRANT OPTION**  
Permite aos usuários a concederem a outros usuários os mesmos privilégios que ele tem.

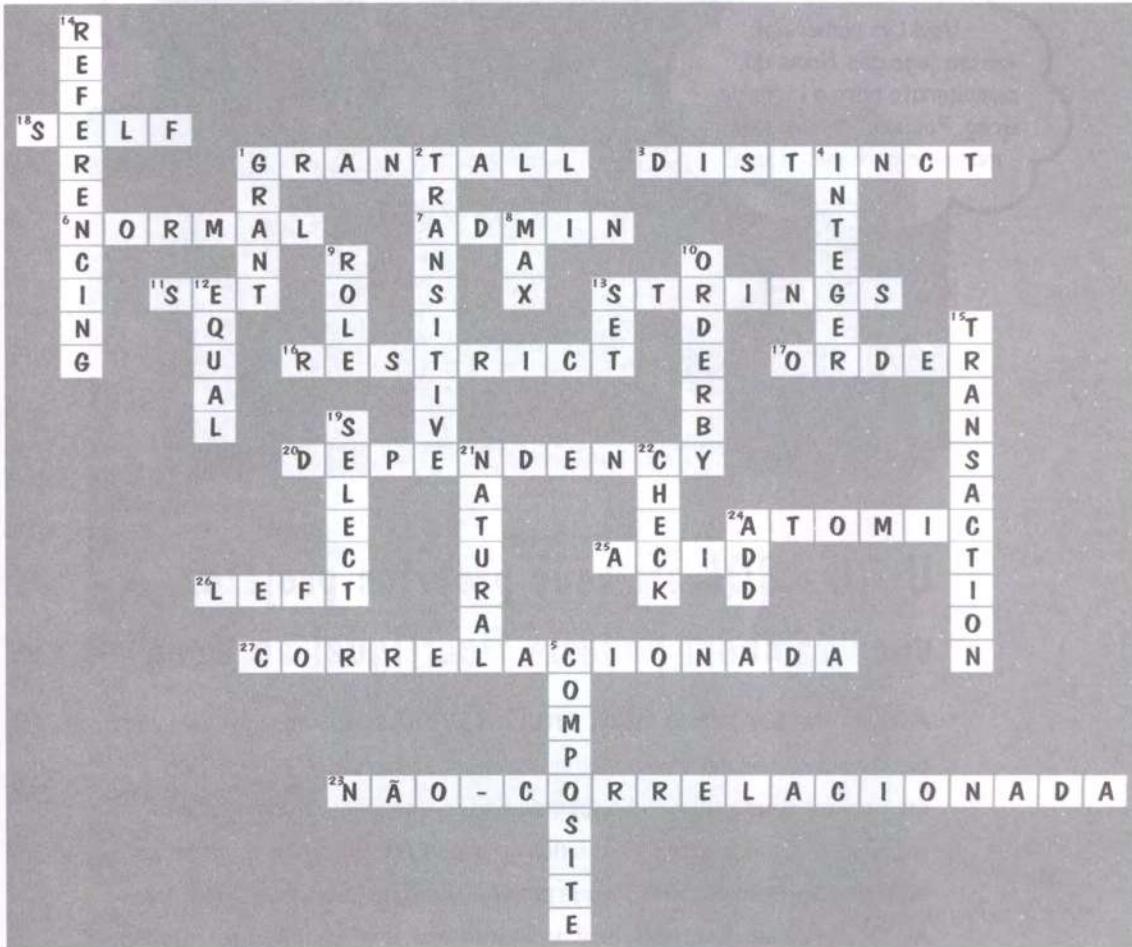
**WITH ADMIN OPTION**  
Permite que qualquer um com uma role conceda aquela mesma role para qualquer outra pessoa.

**GRANT**  
Permite que controle exatamente o que os usuários podem fazer com as tabela e colunas baseado nos privilégios que você dá.

**Role**  
Uma role é um grupo de privilégios. Roles permite que você agrupe privilégios específicos e atribua-os a mais de um usuário.



## (A última) Cruzada SQL - Solução



## O que acha de ter uma Greg's list na sua cidade?



Uau! Um comercial  
em um jogo das finais do  
campeonato para a Lista de  
Greg. Foi uma longa viagem,  
mas olhe para mim agora!

### Utilize SQL em seus próprios projetos, e você também poderia ficar igual ao Greg

Amamos ter você aqui em Dataville, e estamos tristes de vê-lo partir, mas não há nada como *tomar o que você aprendeu* e **colocar isto em uso** em seus próprios bancos de dados – estamos certos de que há palhaços que precisam ser rastreados ou donuts que precisam ser testados, ou [insira seu nome aqui]'s List que precisa de criação onde quer que esteja. Ainda há algumas jóias para você no final do livro, um índice para ler e, então, é hora de pegar todas essas novas idéias e colocá-las em prática. Estamos ansiosos para ouvir como as coisas estão indo, então entre em contato através do website Use a Cabeça Labs, [www.headfirstlabs.com](http://www.headfirstlabs.com), e nos diga como SQL está recompensando VOCÊ!

## apêndice i: sobras

# Os Tópicos Top 10 (que não cobrimos)



**Mesmo depois de tudo isso, ainda há mais um pouco.** Há apenas mais algumas coisas que acreditamos que você precisa saber. Não nos sentiríamos bem em ignorá-las, ainda que elas precisam de apenas uma pequena menção. Então antes de encostar o livro, leia **estas pequenas, mas importantes migalhas**.

## Nº. 1 Pegue um GUI para seu Sistema

Enquanto é importante estar apto a codificar seu SQL diretamente em um console, você sabe o que está fazendo agora. Você merece uma maneira mais fácil de criar suas tabelas e ver seu conteúdo.

Cada Sistema SQL tem algum tipo de interface gráfica associada a ele. Aqui está um breve resumo das ferramentas GUI disponíveis para MySQL.

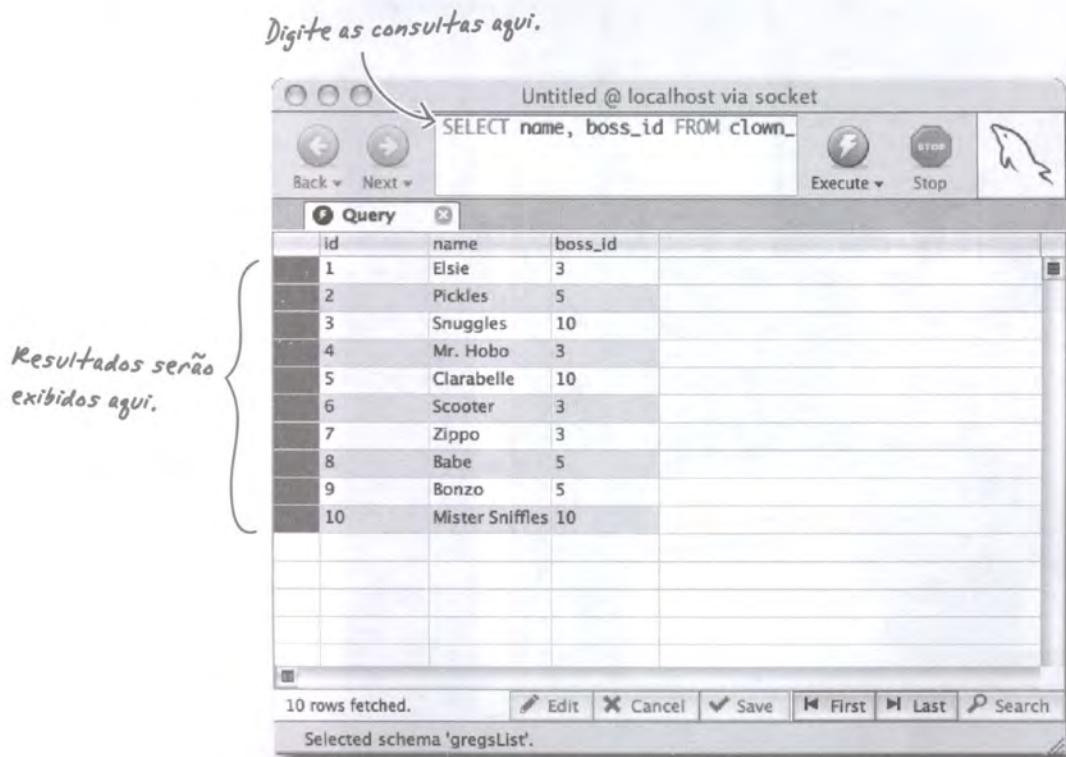
### Ferramentas GUI MySQL

Quando você baixa o MySQL, pode também baixar as ferramentas MySQL GUI, e mais importante, MySQL Administrator. Você pode conseguir a coleção diretamente desta página:

<http://dev.mysql.com/downloads/gui-tools/5.0.html>

Ele está disponível para Windows, Mac e Linux. O MySQL Administrator permite que você veja, crie e modifique seus bancos de dados e tabelas.

Você também vai gostar do MySQL Query Browser. Lá poderá digitar suas consultas e ver o resultado dentro de uma interface do software, ao invés de uma janela do console.

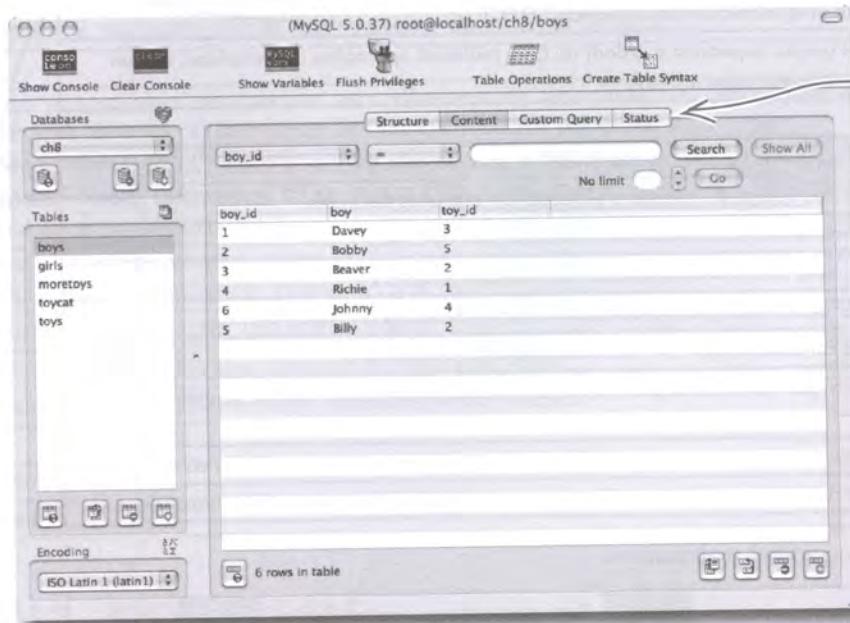


### Outras ferramentas GUI

Há uma porção de outras opções por aí. Deixaremos por sua conta qual você deve escolher. Há muitas ainda nem citadas aqui, as quais você poderá facilmente encontrar fazendo uma pesquisa na web.

Para Mac, você pode tentar CocoaMySQL:

<http://cocoamysql.sourceforge.net/>



Vê a estrutura facilmente, executa uma consulta e altera sua tabela através destes botões.

caso precise de uma solução em web, tente phpMyAdmin. Ele funciona bem se estiver fazendo uma conta de hospedagem com MySQL em um servidor web remoto. Não é tão bom se for utilizando sua própria máquina. Maiores informações podem ser encontradas aqui:

<http://www.phpmyadmin.net/>

estão mais algumas ferramentas comumente utilizadas. Algumas são apenas para PC; seu melhor palpite seria visitar sites e ler suas últimas informações sobre publicações para descobrir se servirão.

oferece 30 dias de teste grátis

<http://www.navicat.com/>

Lyog oferece uma Edição para Comunidade aqui:

<http://www.webbyog.com/en/>

## 1.2 Palavras Reservadas e Caracteres Especiais

Linguagem SQL consiste de palavras-chave reservadas. É melhor manter estas palavras longe do nome de seu banco de dados, tabela ou coluna. Ainda que goste de nomear sua nova tabela de "select" tente criar algum nome mais descriptivo, que não utiliza a palavra "select" de forma alguma. Se você deve utilizar uma palavra chave, tente utilizá-la com outras palavras ou com sublinhadas para que seu Sistema não se confunda. Para sua conveniência, na página à direita está uma lista daquelas palavras reservadas que você vai querer evitar em seus nomes:

assuntos mais complicados, SQL tem uma lista de palavras não-reservadas que podem se tornar reservadas em futuras atualizações do SQL. Não iremos listá-las, mas você pode encontrá-las no aquele livro de referência que deverá comprar quando terminar este livro.

### Caracteres especiais

Aqui está uma lista da maioria de caracteres SQL utilizada e quando elas são utilizadas. Tal forma como as palavras reservadas, é melhor evitar usar estes em seus nomes com a exceção da sublinha (\), a qual nós encorajamos que utilize nos seus nomes. Em geral, é melhor evitar qualquer coisa, exceto letras e sublinhadas no nome de suas tabelas. Números também não são uma grande idéia, a não ser que eles sejam, de alguma forma, descriptivos também.

*	Retorna todas as colunas em uma tabela de um comando SELECT
()	Utilizada por expressões em grupo, especifica a ordem na qual realizará operações matemáticas, e fazer ligações. Também usada para conter subconsultas.
;	Termina seu comando SQL.
,	Separa uma lista de itens. Seu uso inclui o comando INSERT e uma cláusula IN.
.	Usada para fazer referência aos nomes de tabelas e utilizada em números decimais.
-	Este é um coringa que representa um só caractere em uma cláusula LIKE.
%	Outra cláusula LIKE coringa, esta aqui é usada no caso de múltiplos caracteres.
!	O ponto de exclamação quer dizer NÃO. Ela é usada com comparadores na cláusula WHERE.
'	Um par de aspas simples diz ao Sistema que um valor de texto está entre eles.
"	Você também pode utilizar um par de aspas duplas da mesma forma, embora seja melhor se ater a forma de aspas simples.
\	Este caractere é usado para lhe permitir colocar uma aspa simples dentro de um texto na sua coluna ou sua tabela.
+	Além de utilizá-lo como sinal de adição, você pode utilizá-lo para conectar ou concatenar dois faixas de textos.

Aqui está uma olhada rápida nos operadores matemáticos:

+	Adição	+	Subtração	*	Entre dois valores, o asterisco age como símbolo de multiplicação.	+	Divisão
---	--------	---	-----------	---	--	---	---------

And the comparison operators:

>	Maior que	!>	Não maior que	>=	Maior ou igual
<	Menor que	!>	Não menor que	>=	Menor ou igual
=	Igual a	↔	Não igual	!=	Não igual

&		^	~
Não abordado neste livro. Cheque seu manual do sistema SQL para maiores informações.			

## Palavras reservadas

É uma ótima ideia sempre olhar estas palavras quando quer que esteja nomeando uma tabela, para certificar-se de que não está utilizando nenhuma destas palavras.

A	ABSOLUTE ACTION ADD ADMIN AFTER AGGREGATE ALIAS ALL ALLOCATE ALTER AND ANY ARE ARRAY AS ASC ASSERTION AT AUTHORIZATION
B	BEFORE BEGIN BINARY BIT BLOB BOOLEAN BOTH BREADTH BY
C	CALL CASCADED CASE CAST CATALOG CHAR CHARACTER CHECK CLASS CLOB CLOSE COLLATE COLLATION COLUMN COMMIT COMPLETION CONNECT CONNECTION CONSTRAINT CONSTRAINTS CONSTRUCTOR CONTINUE CORRESPONDING CREATE CROSS CUBE CURRENT CURRENT_DATE CURRENT_PATH CURRENT_ROLE CURRENT_TIME CURRENT_TIMESTAMP CURRENT_USER CURSOR CYCLE
D	DATA DATE DAY DEALLOCATE DEC DECIMAL DECLARE DEFAULT DEFERRABLE DEFERRED DELETE DEPTH DEREF DESC DESCRIBE DESCRIPTOR DESTROY DESTRUCTOR DETERMINISTIC DICTIONARY DIAGNOSTICS DISCONNECT DISTINCT DOMAIN DOUBLE DROP DYNAMIC
E	EACH ELSE END END_EXEC EQUALS ESCAPE EVERY EXCEPT EXCEPTION EXEC EXECUTE EXTERNAL
F	FALSE FETCH FIRST FLOAT FOR FOREIGN FOUND FROM FREE FULL FUNCTION
G	GENERAL GET GLOBAL GO GOTO GRANT GROUP GROUPING
H	HAVING HOST HOUR
I	IDENTITY IGNORE IMMEDIATE IN INDICATOR INITIALIZE INITIALLY INNER INOUT INPUT INSERT INT INTEGER INTERSECT INTERVAL INTO IS ISOLATION ITERATE
J	JOIN
K	KEY
L	LANGUAGE LARGE LAST LATERAL LEADING LEFT LESS LEVEL LIKE LIMIT LOCAL LOCALTIME LOCALTIMESTAMP LOCATOR
M	MAP MATCH MINUTE MODIFIES MODIFY MODULE MONTH
N	NAMES NATIONAL NATURAL NCHAR NCLOB NEW NEXT NO NONE NOT NULL NUMERIC
O	OBJECT OF OFF OLD ON ONLY OPEN OPERATION OPTION OR ORDER ORDINALITY OUT OUTER OUTPUT

P	PAD PARAMETER PARAMETERS PARTIAL PATH POSTFIX PRECISION PREFIX PREORDER PREPARE PRESERVE PRIMARY PRIOR PRIVILEGES PROCEDURE PUBLIC
Q	
R	READ READS REAL RECURSIVE REF REFERENCES REFERENCING RELATIVE RESTRICT RESULT RETURN RETURNS REVOKE RIGHT ROLE ROLLBACK ROLLUP ROUTINE ROW ROWS
S	SAVEPOINT SCHEMA SCROLL SCOPE SEARCH SECOND SECTION SELECT SEQUENCE SESSION SESSION_USER SET SETS SIZE SMALLINT SOME SPACE SPECIFIC SPECIFICTYPE SQL SQLEXCEPTION SQLSTATE SQLWARNING START STATE STATEMENT STATIC STRUCTURE SYSTEM_USER
T	TABLE TEMPORARY TERMINATE THAN THEN TIME TIMESTAMP TIMEZONE_HOUR TIMEZONE_MINUTE TO TRAILING TRANSACTION TRANSLATION TREAT TRIGGER TRUE
U	UNDER UNION UNIQUE UNKNOWN UNNEST UPDATE USAGE USER USING
V	VALUE VALUES VARCHAR VARIABLE VARYING VIEW
W	WHEN WHENEVER WHERE WITH WITHOUT WORK WRITE
X	
Y	YEAR
Z	ZONE

## 10.3 ALL, ANY e SOME

As palavras-chave que são uma mão na roda com subconsultas. Estas são a ALL, ANY e SOME. Elas trabalham com operadores de comparação e conjuntos de resultados. Antes de escolhermos, vamos dar uma espiada no operador IN sobre o qual conversamos no Capítulo 9.

Nota\_restaurantes

Nome	nota
Pizza House	3
The Shack	7
Arthur's	9
Ribns' n' More	5

```
SELECT nome, nota FROM nota_restaurante
WHERE nota IN
(SELECT nota FROM nota_restaurante ← A subconsulta retorna qualquer nota entre
WHERE nota > 3 AND nota < 9);            3 e 9; neste caso, 7 e 5.
```

A consulta retorna o nome de qualquer restaurante com a mesma nota que o resultado de nossa subconsulta no conjunto entre parênteses. Nossos resultados serão **The Shack** e **Ribs' n' More**.

### Utilizando o ALL

Agora considere esta consulta:

```
SELECT nome, nota FROM nota_restaurante
WHERE nota > ALL
(SELECT nota FROM nota_restaurante
WHERE nota > 3 AND nota < 9);
```

Essa vez iremos pegar todos os restaurantes com as maiores notas de todas as notas em seu conjunto. Nossa resultado será **Arthur's**.

Essa é uma consulta com <:

```
SELECT nome, nota FROM nota_restaurante
WHERE nota < ALL
(SELECT nota FROM nota_restaurante
WHERE nota > 3 AND nota < 9);
```

Maior que ALL (maior que todos) encontra qualquer valor maior que o maior valor no conjunto.

Menor que ALL (menor que todos) encontra qualquer valor menor que o menor valor no conjunto.

Podemos utilizar também `>=` e `<=` com ALL. Esta consulta nos dará os resultados **Pizza Shack** e **Ribs`n`More**. Nós obtivemos os maiores resultados que nosso conjunto, bem como todos que são iguais a maior que temos no nosso conjunto, que é 7:

```
SELECT nome, nota FROM nota_restaurante
WHERE nota >= ALL
(SELECT nota FROM nota_restaurante
WHERE nota > 3 AND nota < 9);
WHERE rating > 3 AND rating < 9;
```

*Qualquer valor maior que nosso conjunto ou igual ao maior resultado de nosso conjunto será combinado.*

## Usando ANY (qualquer)

ANY avalia como verdadeiro se qualquer valor do conjunto combinar com a condição. Observe o seguinte exemplo:

```
SELECT nome, nota FROM nota_restaurante
WHERE nota > ANY
(SELECT nota FROM nota_restaurante WHERE
nota > 3 AND nota < 9);
```

Podemos ler isto como: selecione qualquer linha onde a nota é maior que qualquer de (5,7). Uma vez que **The Shack** tem a nota 7, que é maior que 5, ela é retornada. E **Arthur's** que tem nota 9, também é retornado.

## Usando SOME

SOME quer dizer que a mesma coisa que ANY na sintaxe padrão e no MySQL. Cheque seu sistema de preferência para confirmar que esta palavra-chave funciona da mesma forma para você.

Maior que ANY (maior que qualquer) encontra qualquer valor maior que o menor valor no conjunto.

Menor que ANY (menor que qualquer) encontra qualquer valor menor que o maior valor no conjunto.

## Nº. 4 Mais Tipos de Dados

Você conhece os tipos de dados mais comuns, mas há alguns detalhes que podem ajudar a deixar suas colunas ainda mais refinadas. Vamos olhar de perto para alguns novos tipos de dados, e um olhar mais aprofundado naqueles que já estamos usando.

### BOOLEAN

O tipo de dados booleano permite que armazene ‘verdadeiro’ ou ‘falso’, ou então pode ser deixado como valor NULL. É ótimo para qualquer tipo de coluna verdadeiro / falso. Por detrás das cortinas, seu Sistema SQL está armazenando 1 para valores verdadeiros e 0 para valores falsos. Você pode inserir 1 ou “true”, 0 ou “false”.

### INT

Nós temos utilizado INT ao longo de todo o livro. INT pode armazenar **valores na faixa de 0 a 4294967295**. Isto se você quiser utilizar apenas valores positivos, e isto é conhecido como **integer não assinada**.

Se quiser utilizar valores positivos e negativos em sua integer, você precisa fazer dele uma **integer assinada**. Uma integer assinada pode armazenar valores de -2147483648 a 2147483647. Para dizer ao seu Sistema SQL que você quer uma INT assinada, utilize esta sintaxe quando a criar:

**INT(SIGNED)**

## outros tipos de dados INT

Já conhece INT, mas os dois tipos de dados SMALLINT e BIGINT a exatamente o mesmo. Eles especificam um valor máximo que pode ser armazenado.

Faixas de valores que elas podem armazenar variam de acordo com seu Sistema MySQL. As faixas do alcance do MySQL são:

	signed	unsigned
SMALLINT	-32768 to 32767	0 to 65535
BIGINT	-9223372036854775808 to 9223372036854775807	0 to 18446744073709551615

MySQL vai mais longe e adiciona mais os dois tipos de dados abaixo:

	signed	unsigned
TINYINT	-128 to 127	0 to 255
MEDIUMINT	-8388608 to 8388607	0 to 16777215

## outros tipos de dados DATE e TIME

Está um resumo do formato no qual MySQL armazena seus tipos de dados de hora e data:

DATE	<code>YYYY-MM-DD</code>	Algumas_datas
DATETIME	<code>YYYY-MM-DD HH:MM:SS</code>	Uma_data
TIMESTAMP	<code>YYYYMMDDHHMMSS</code>	2007-08-25 22:10:00
TIME	<code>HH:MM:SS</code>	1925-01-01 02:05:00

Quando utiliza o tipo date ou time, você pode modificar o que seu Sistema exibe. Funções fazem isso de acordo com o Sistema. Aqui está um exemplo da função no MySQL `DATE_FORMAT()` que responde que você tivesse a coluna `uma_data`:

```
SELECT DATE_FORMAT(uma_data, '%M %Y') FROM algumas_datas;
```

%M e %Y dizem para a função como você quer o formato das datas. Aqui está como o resultado seria:

Uma_data
Agosto 2007
Janeiro 1925

Temos espaço aqui para apresentar todas as opções de formato; há um grande número delas. Com elas, você pode obter exatamente o que precisa do seu campo data e hora, sem ter que ver que não precisa.

## Nº. 5 Tabelas temporárias

Criamos várias tabelas neste livro. Cada vez que criamos uma tabela, nosso Sistema SQL armazena a estrutura daquela tabela. Quando inserirmos dados nela, aqueles dados serão armazenados. A tabela e os dados contidos nela são salvos. Se você se desconectar da sua sessão SQL na sua janela de terminal ou deletá-la; a tabela persistirá até que utilize o `DROP`.

SQL oferece outro tipo de dados, conhecido como tabela temporária. Uma **tabela temporária** existe desde o momento que você a criou até o momento de eliminá-la, *ou até a sessão ser finalizada*. Por **sessão**, queremos dizer o tempo que você permaneceu conectado a sua conta até ter realizado o encerramento da conexão ou finalizado seu programa GUI. Você pode também ter eliminado explicitamente como o comando `DROP`.

### Razões pelas quais você pode querer uma tabela temporária:

- Você pode usá-la para armazenar resultados intermediários – por exemplo, realizando alguma operação matemática em uma coluna. Os resultados daqueles que precisará reutilizar durante a sessão, mas não até a próxima sessão.
- Você quer capturar o conteúdo de uma tabela em um momento específico.
- Ainda se lembra de quando convertemos a Greg's List um relacionamento um-para-muitos? Você pode criar tabelas temporárias para ajudá-lo a reestruturar seus dados, e saber que elas podem ser eliminadas quando você tiver finalizado sua sessão.
- Se você eventualmente utilizar SQL com uma linguagem de programação, pode criar tabelas temporárias para juntar os dados depois armazenar os resultados finais em uma tabela permanente.

### Crie uma tabela temporária

A sintaxe para criar uma tabela temporária em MySQL é simples, você adicionar a palavra-chave `TEMPORARY`:

```
CREATE TEMPORARY TABLE minha_tabela_temporaria
(
    coluna_id INT,
    algum_dado VARCHAR(50)
)
```

*A palavra TEMPORARY é a única coisa que precisamos adicionar.*



**Veja Isto!**

**A sintaxe para criação de tabelas pode variar grandemente de um Sistema para outro**

*Certifique-se de verificar na documentação de seu Sistema sobre este recurso.*

### Um atalho para uma tabela temporária

Você pode criar sua tabela temporária a partir de uma consulta como esta:

```
CREATE TEMPORARY TABLE minha_tabela_temporaria AS
SELECT * FROM minha_tabela_permanente;
```

*Qualquer consulta que você quiser vai irá depois de AS.*

## Nº. 6 Molde seus dados

Às vezes você tem uma coluna de um certo tipo de dados, mas quer que ela seja de um tipo diferente de dados ao ser exibida. SQL tem uma função chamada `CAST ()` que pode fazer dados de um tipo serem convertidos em outro.

A sintaxe é esta:

```
CAST (sua_coluna, TIPO)
```

O pode ser qualquer um destes:

```
CHAR()
DATE
DATETIME
DECIMAL
SIGNED [INTEGER]
TIME
UNSIGNED [INTEGER]
```

### Almas situações onde você pode querer utilizar o comando CAST()

Inverte uma linha de texto com data em tipo DATE:

```
SELECT CAST('2005-01-01' AS DATE);
```

← A linha de texto '2005-01-01' é formatada como uma DATE.

Inverte um tipo integer para decimal:

```
SELECT CAST(2 AS DECIMAL);
```

← A integer (número inteiro) 2 se torna decimal 2.00.

Outros lugares que você pode utilizar o CAST() incluem a lista de valores de um comando INSERT e dentro da lista de coluna de um comando SELECT.

### Não pode utilizar CAST() nestas situações

Decimal para integer

TIME, DATE, DATETIME, CHAR para DECIMAL, ou INTEGER.

Em alguns outros lugares você pode utilizar CAST(), incluir a lista de valores de um comando INSERT e dentro da lista de coluna do SELECT.

## 1.7 Quem é você? Que horas são?

Muitas vezes você pode ter mais que uma conta de usuário no seu Sistema SQL, cada uma com diferentes permissões e roles. Se precisar saber qual conta você está atualmente usando, este comando irá informá-lo:

```
SELECT CURRENT_USER;
```

E também dirá qual é a máquina host. Se seu Sistema SQL está no mesmo computador que estiver utilizando, e for uma conta raiz, você verá isto:

```
root@localhost
```

Você pode obter a data e hora atual com estes comandos:

```
File Edit Window Help
> SELECT CURRENT_DATE;
+-----+
| CURRENT_DATE |
+-----+
| 2007-07-26 |
+-----+
1 row in set (0.00 sec)
```

```
File Edit Window Help
> SELECT CURRENT_TIME;
+-----+
| CURRENT_TIME |
+-----+
| 11:26:48 |
+-----+
1 row in set (0.00 sec)
```

```
File Edit Window Help
SELECT CURRENT_USER;
+-----+
| CURRENT_USER |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)
```

## Nº. 8 Funções numéricas úteis

Aqui está um resumo das funções que funcionam com tipos de dados numéricos. Alguns você já conhece:

Função numérica	o que ela faz?	
<b>ABS (x)</b>	Retorna o valor absoluto de x	
	Consulta	Resultado
	SELECT ABS (-23) ;	23
<b>ACOS (x)</b>	Retorna o arco co-seno de x	
	SELECT ACOS (0) ;	1.5707963267949
<b>ASIN ()</b>	Retorna o arco seno x	
	SELECT ASIN(0.1) ;	0.10016742116156
<b>ATAN (x, y)</b>	Retorna o arco tangente de x e y	
	SELECT ATAN(-2,2) ;	-0.78539816339745
<b>CEIL (x)</b>	Retorna o menor valor inteiro que é maior ou igual a x. O valor retornado será uma BIGINT.	
	SELECT CEIL(1.32) ;	2

Função numérica	o que ela faz?	
	Consulta	Resultado
COS (x)	Retorna o co-seno de x em radiano SELECT COS(1);	0.54030230586814
COT (x)	Retorna a co-tangente de x SELECT COT(12);	-1.5726734063977
EXP (x)	Retorna o valor de e elevado ao exponencial de x SELECT EXP(-2);	0.13533528323661
FLOOR (x)	Retorna o maior valor inteiro que é menor ou igual a x SELECT FLOOR(1.32);	1
FORMAT (x, y)	Converte x em uma linha de texto formatada e arredonda por casas decimais SELECT FORMAT(3452100.50, 2);	3,452,100.50
LN (x)	Retorna o logaritmo natural de x SELECT LN(2);	0.69314718055995
LOG (x) e LOG (x, y)	Retorna o logaritmo natural de x, ou com dois parâmetros retorna o logaritmo de x para base y SELECT LOG(2); SELECT LOG(2, 65536);	0.69314718055995 16
MOD (x, y)	Retorna o remanescente de x dividido por y SELECT MOD(249, 10);	9
PI ()	Retorna o valor de pi SELECT PI();	3.141593
POWER (x, y)	Retorna o valor de x elevado ao exponencial de y SELECT POW(3, 2);	9
RADIANS (x)	Retorna x convertido de graus para radianos SELECT RADIANS(45);	0.78539816339745
RAND ()	Retorna um ponto flutuante aleatório SELECT RAND();	0.84655920681223
ROUND (x)	Retorna o valor de x arredondado para o número inteiro mais próximo SELECT ROUND(1.34); SELECT ROUND(-1.34);	1 -1
ROUND (x, y)	Retorna o valor de x arredondado para as casas decimais de y SELECT ROUND(1.465, 1); SELECT ROUND(1.465, 0); SELECT ROUND(28.367, -1);	1.5 1 30
SIGN (x)	Retorna 1 quando x é positivo, 0 quando x é 0, ou -1 quando x é negativo SELECT SIGN(-23);	-1
SIN (x)	Retorna o seno de x SELECT SIN(PI());	1.2246063538224e-16
SQRT (x)	Retorna a raiz quadrada de x SELECT SQRT(100);	10
TAN (x)	Retorna a tangente de x SELECT TAN(PI());	-1.2246063538224e-16
TRUNCATE (x, y)	Retorna o número x truncado em decimais de y SELECT TRUNCATE(8.923, 1);	8.9

## Nº. 9 Indexando para deixar as coisas mais rápidas

Você sabe tudo sobre indexação com chaves primárias e chaves externas. Estes tipos de índices são ótimos para amarrar tabelas e forçar a integridade dos dados. Mas você pode criar também índices para colunas para tornar suas consultas mais rápidas.

Quando uma cláusula WHERE está pronta em uma coluna não-indexada, o Sistema SQL começa do início da coluna e vai até o final, uma linha de cada vez. Se sua tabela é enorme, e queremos dizer, enorme com 4 milhões de linhas, isto pode ser perceptível ao longo do tempo.

Quando você cria um índice de coluna, seu Sistema SQL mantém informações adicionais sobre as colunas que aceleram tremendamente aquela procura. A informação adicional é mantida em uma tabela por detrás das cortinas que é uma ordem específica onde o Sistema SQL poderá procurar através das colunas mais rapidamente. A contraprestação é que índices ocupam espaço. Então você deve considerar a criação de algumas colunas como índices, aquelas onde fará buscas freqüentemente, e não indexar as demais.

Aqui está o código do comando ALTER TABLE para adicionar um índice em uma coluna:

```
ALTER TABLE meus_contatos
ADD INDEX (sobrenome);
```

Há um pouco mais de teoria por detrás do tema indexação, mas a idéia básica é esta.

## Nº. 10 PHP/MySQL em dois minutos

Antes de terminarmos, vamos dar uma olhada rápida em como PHP e MySQL podem interagir juntas para ajudá-lo a obter seus dados na Web. Esta é apenas uma prova do que pode ser feito, e você com certeza deveria ler mais sobre isso.

Este exemplo presume que você seja um pouco familiar com PHP. Nós sabemos o quanto está confortável em escrever consultas neste ponto. O código abaixo conecta o banco de dados chamada gregs\_list e seleciona todos os primeiros nomes e sobrenomes das pessoas na tabela meus\_contatos. O código PHP pega todos estes dados do banco de dados e armazena em um vetor (array). A última parte do código imprime todos os nomes e sobrenomes em uma página da web:

```
<?php
$conn = mysql_connect("localhost", "greg", "gr3gzpAs");
if (!$conn)
{
    die('Did not connect: ' . mysql_error());
}

mysql_select_db("my_db", $conn);
$result = mysql_query("SELECT primeiro_nome, sobrenome FROM meus_contatos");
while($row = mysql_fetch_array($result))
{
    echo $row['primeiro_nome'] . " " . $row['sobrenome'];
    echo "<br />";
}
mysql_close($conn);
?>
```

Salvaremos o arquivo como gregsnomes.php em nosso servidor web.

## Um olhar aprofundado sobre cada linha

```
<?php
```

Esta linha diz ao Servidor Web que abaixo segue um código PHP.

```
$conn = mysql_connect("localhost", "greg", "gr3gzpAs");
```

Agora para conectar a gregs\_list, temos que dizer onde nosso Sistema SQL está localizado, qual o nosso nome de usuário e nossa senha. Nós criamos uma linha de texto conectora com esta informação e a nomeamos \$conn. A função PHP mysql\_connect () pega aquela informação e vai até o Sistema SQL para ver se consegue estabelecer uma conexão.

```
if (!$conn)
{
    die('Did not connect: ' . mysql_error());
}
```

Se fosse bem sucedido, PHP nos enviaria uma mensagem dizendo porque não poderia conectar ao Sistema SQL e o PHP interromperia o processamento.

```
mysql_select_db("my_db", $conn);
```

Agora, então nossa conexão ao Sistema SQL funciona. Agora temos que dizer ao Sistema SQL qual o banco de dados ao qual estamos interessados. Queremos utilizar o nosso banco de dados favorito: gregs\_list.

```
$result = mysql_query("SELECT primeiro_nome, sobrenome FROM meus_contatos");
```

Selecionamos nosso banco de dados, e estamos conectados, mas não temos consulta alguma. Escrevemos uma e utilizamos a função mysql\_query () para enviá-la ao Sistema. Todas as linhas retornadas são armazenadas em uma array chamada \$result.

```
while($row = mysql_fetch_array($result))
{
```

Agora utilizamos o PHP para pegar todas as linhas de \$result e colocá-las na página da web. Isto é feito por uma while loop, que é levada linha por linha até alcançar o final dos dados.

```
echo $row['primeiro_nome'] . " " . $row['sobrenome'];
echo "<br />";
}
```

Estes dois comandos PHP echo escrevem o nome e o sobrenome de cada linha para a página da web. Uma tag HTML <br> é inserida entre cada linha.

```
close($conn);
```

Quando terminamos de escrever todos os nomes, fechamos a conexão do Sistema SQL. É como encerrar uma sessão no terminal.

```
?>
```

Finalmente nós finalizamos o script PHP.



## apêndice ii: Instalação do MySQL

### Tente você mesmo



**Todos as suas novas habilidades não farão muito sem um lugar para aplicá-las.**

Este apêndice contém instruções para instalar seu próprio Sistema SQL para que você possa trabalhar.

## Comece logo!

Pelo fato de não ser divertido ter um livro de SQL sem estar apto a experimentar você mesmo, aqui vai uma breve introdução sobre como instalar o programa MySQL no Windows, Mac e OS X.

**Nota: Esta seção aborda o Windows 2000, XP ou Windows Server 2003 ou outro Sistema Operacional Windows de 32-bit. Para Mac, ele aborda o Mac OS 10.3.x ou mais novo.**

Guaremos você desde baixar o programa até sua instalação. O nome oficial para a versão gráts do servidor Sistema MySQL hoje em dia é **MySQL Community Server**.

## Instruções e Solução de Problemas

A seguir está uma lista de passos para instalação do MySQL para Windows e Mac OS X. Ele não foi criado para substituir as ótimas instruções encontradas no site do próprio sistema, e **nós encorajamos você veementemente a acessar o site e ler estas instruções!** Para direções muito mais detalhadas, bem como o guia de solução de problemas, vá aqui:

↓ Pegue a versão 5.0 ou mais nova.

<http://dev.mysql.com/doc/refman/5.0/en/windows-installation.html>

Você também vai gostar do MySQL Query Browser sobre o qual falamos nas páginas 428/429. Nele, você poderá digitar suas consultas e ver os resultados dentro da interface do próprio software, ao invés de uma janela do console.

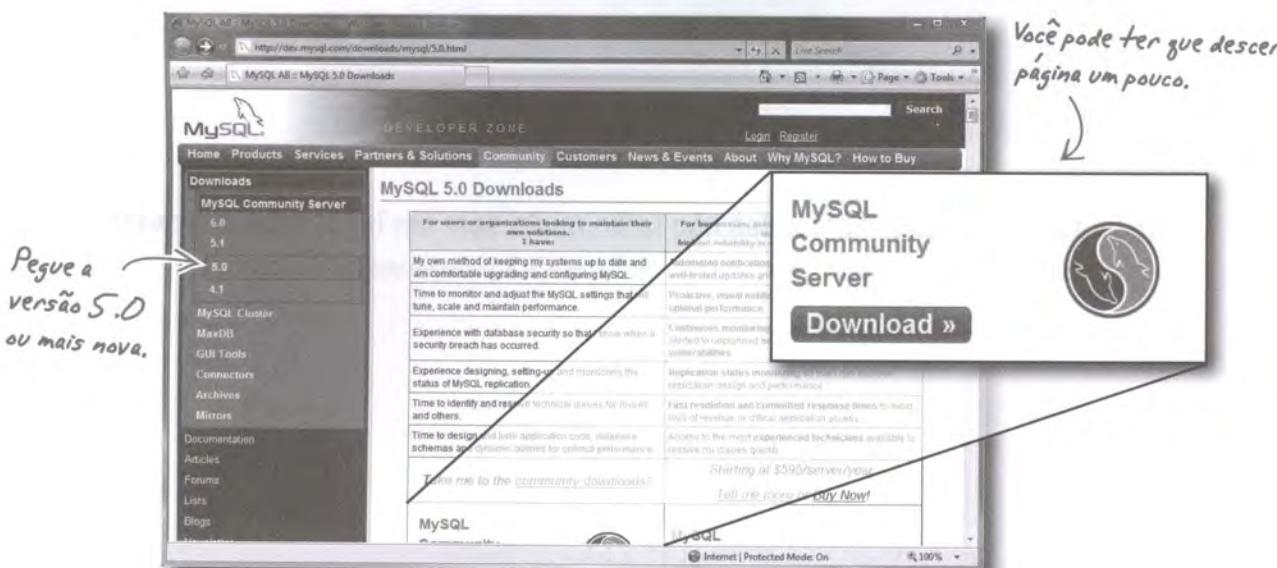
## Passos para instalar MySQL no Windows

1

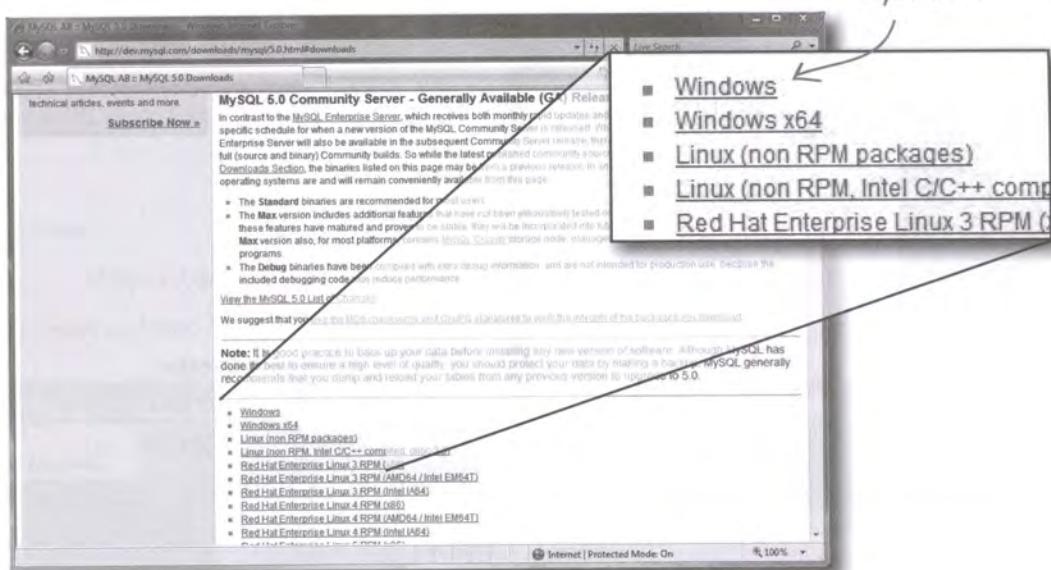
Vá em

<http://dev.mysql.com/downloads/mysql/5.0.html>

e clique no botão MySQL Community Server download.

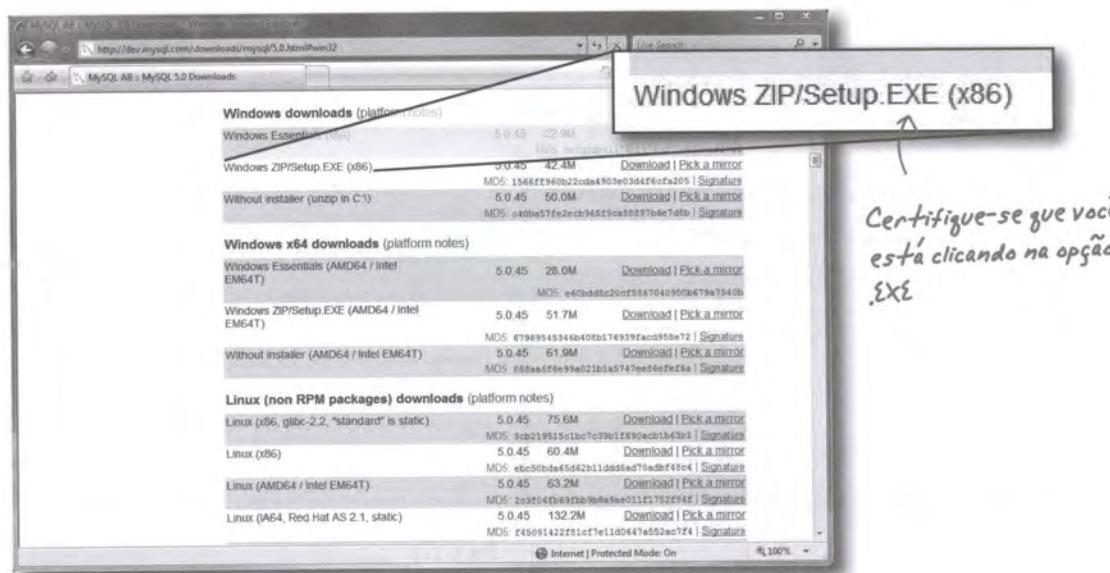


- 2** Escolha **Windows** na lista.



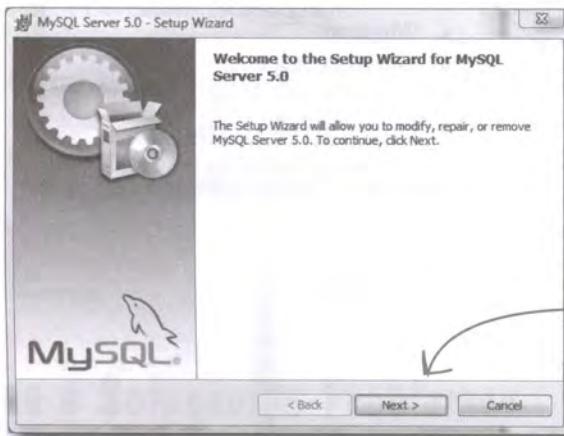
### Baixe seu instalador

- 3** Abaixo de **windows downloads**, recomendamos que você escolha a opção **Windows ZIP / Setup.EXE** porque ela inclui um instalador que simplifica grandemente a instalação. Clique em **Pick a Mirror**.



- 4** Você verá uma lista de localidades que possuem uma cópia que possa baixar, escolha aquela mais próxima de você.

- 5 Quando o arquivo terminar de ser baixado, dê um duplo clique sobre ele para executá-lo. Neste ponto, você será conduzido através da instalação pelo Assistente de Instalação. Clique no botão **next, ou próximo**.



Após ter clicado duas vezes no arquivo e o Assistente de Instalação aparecer, clique no botão **Next**.

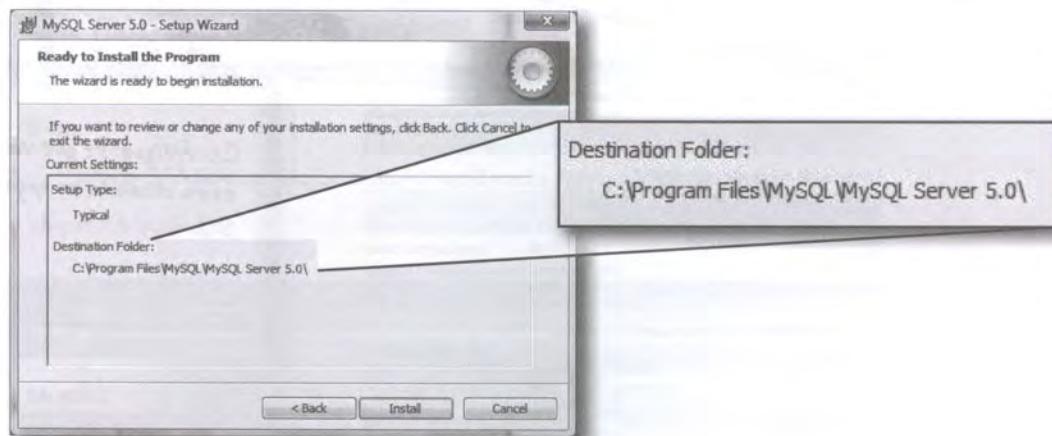
## Escolha uma pasta de destino

- 6 Será pedido para que você escolha entre Típica, Completa ou Padronizada. Para nossos propósitos neste livro, escolha **Típica**.

Você pode alterar o local em seu computador onde o MySQL será instalado, mas recomendamos que deixe no local padrão:

C:\Program Files\MySQL\MySQL Server 5.0

Clique no botão **Next**.



Clique em "install" (instalar) e estará pronto.

- 7 Você verá a caixa de diálogo "Ready to Install" (pronto para instalar) com o local de destino listado. Se estiver satisfeito com o diretório de destino, clique **Install** (instalar). Caso contrário, clique em **Back, altere** o diretório e volte para este ponto.

Clique **Install**.

## Passos para instalar MySQL em Mac OS X

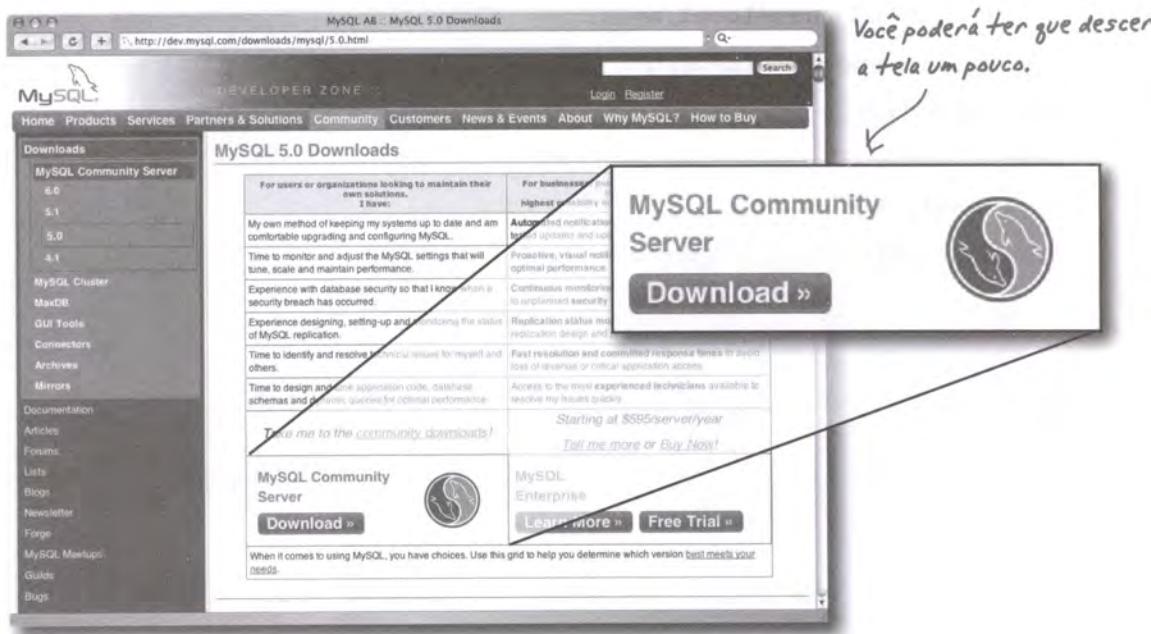
Se está executando um servidor Mac OS X, uma versão do MySQL já deveria estar instalada.

Antes de iniciar, cheque para verificar se você já tem uma versão instalada. Vá em Applications/Server/MySQL Manager para acesso.

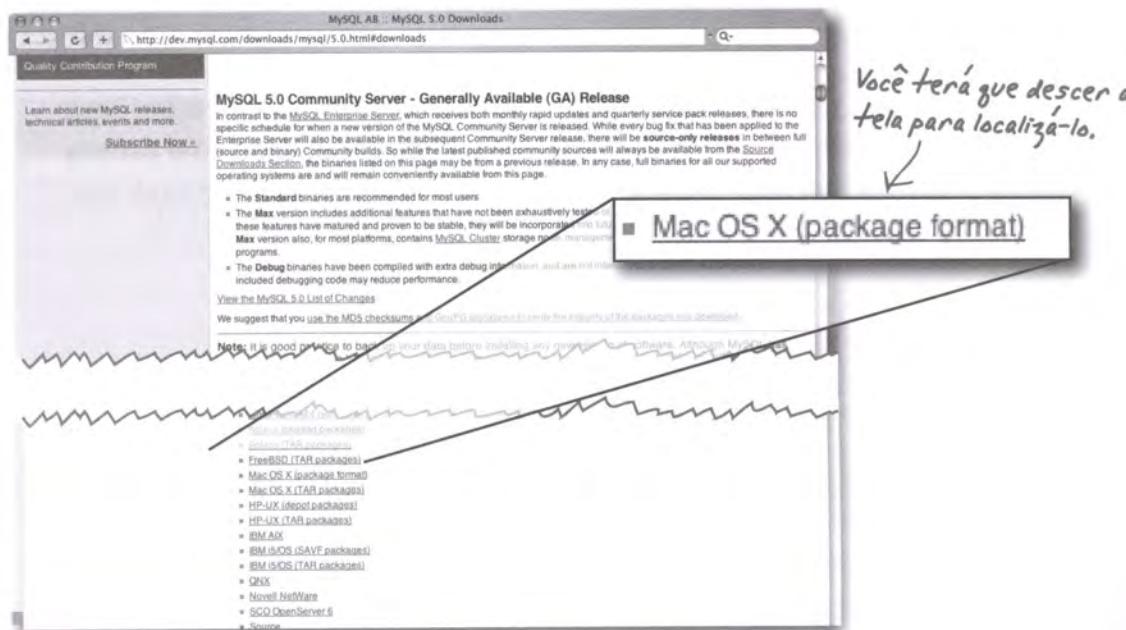
- 1** Vá em:

<http://dev.mysql.com/downloads/mysql/5.0.html>

e clique no botão MySQL Community Server.



- 2** Escolha Mac OS X (“package format”) (formato pacote) constante da lista.



- 3 Escolha o pacote apropriado para sua versão do Mac OS X  
Clique em **Pick a Mirror**.
- 4 Você verá uma lista de locais que possuem uma cópia que possa baixar;  
escolha a localidade mais próxima de você.
- 5 Quando o arquivo terminar o download, dê dois cliques para executá-lo. Quando você tiver instalado o MySQL vá verificar documentações online para como ter acesso a sua instalação utilizando o query browser sobre o qual falamos nas páginas 428/429.

Mas se estiver com pressa, aqui está uma forma rápida de utilizar o Terminal.

Você agora pode abrir uma janela terminal no seu Mac e digitar:

```
shell> cd /usr/local/mysql  
shell> sudo ./bin/mysqld_safe
```

(Insira sua senha, se necessário)

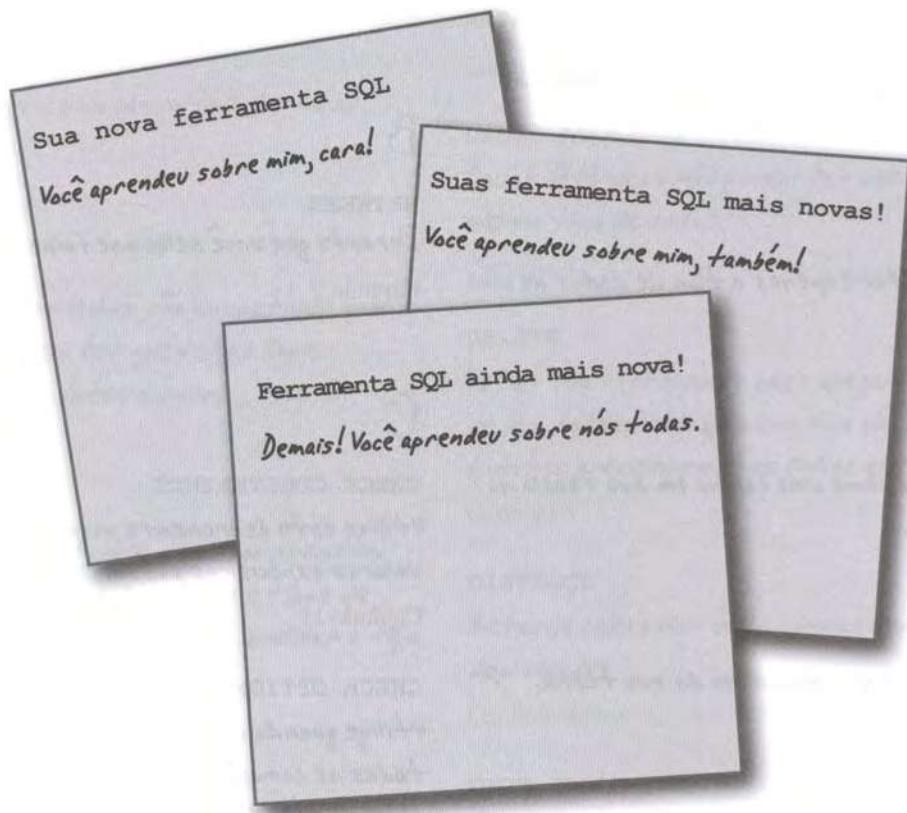
(Aperte Control-Z)

```
shell> bg
```

(Aperte Control-D ou aperte Exit para sair da shell)

## apêndice iii: compilação de ferramentas

### Todas as suas novas ferramentas SQL



Aqui todas as suas ferramentas SQL estão em um só lugar pela primeira vez, por apenas uma noite (**brincadeirinha**)! Esta é uma compilação de todas as ferramentas que abordamos. Gaste alguns minutos para analisar a lista e poder se sentir o máximo – pois você aprendeu todas elas!!!

## Símbolos

= <> < > <= >=

Você tem uma porção de operadores de igualdade e desigualdade à sua disposição.

Capítulo 2

## A

ALTER c\ CHANGE

Permite que você altere o nome e o tipo de dados de uma coluna existente.

Capítulo 5

ALTER c\ MODIFY

Permite que você altere apenas o tipo de dados de uma coluna existente.

Capítulo 5

ALTER c\ ADD

Permite que você adicione uma coluna em sua tabela na ordem que escolher.

Capítulo 5

ALTER c\ DROP

Permite que você elimine uma coluna da sua tabela.

Capítulo 5

ALTER TABLE

Permite que altere o nome de sua tabela ou da estrutura inteira enquanto retém os dados dentro dela.

Capítulo 5

AND e OR

Com AND e OR, você pode combinar seus comandos condicionais nas suas cláusulas WHERE para maior precisão.

Capítulo 2

AUTOCONEXÃO (SELF-JOIN)

A autoconexão permite que você consulte uma só tabela embora existam duas tabelas com exatamente as mesmas informações nela inseridas.

Capítulo 10

AUTO\_INCREMENT

Quando utilizada na declaração de sua coluna, aquela coluna irá atribuir um número inteiro automaticamente, cada vez que um comando INSERT for realizado.

Capítulo 4

AVG

Retorna o valor médio de uma coluna numérica.

Capítulo 6

## B

BETWEEN

Permite que você selecione faixa de valores.

Capítulo 2

## C

CHECK CONSTRAINTS

Utilize esta ferramenta para permitir que apenas valores específicos possam ser inseridos.

Capítulo 11

CHECK OPTION

Utilize quando criar uma view atualizável para forçar todos os comandos inserts e updates a satisfazer a cláusula WHERE na view.

Capítulo 11

COMMA JOIN

A mesma coisa que CONEXÃO CRUZADA (CROSS JOIN), exceto que a vírgula é utilizada ao invés das palavras-chave CROSS JOIN.

Capítulo 8

CHAVE COMPOSTA (COMPOSITE KEY)

Esta é uma chave primária feita de múltiplas colunas que criaram um valor de chave único.

Capítulo 7

Chave Estrangeira (Foreign Key)

Uma coluna em uma tabela que se baseia em uma chave primária de uma outra tabela.

Capítulo 6

**CHAVE Estrangeira AUTO-REFERENCIADA**

Ista é uma chave estrangeira na mesma tabela que a chave primária, mas utilizada para outro propósito.

Capítulo 10

**CHAVE PRIMÁRIA**

Uma coluna ou conjunto de colunas que identificam uma linha de dados em uma tabela.

Capítulo 4

**Consulta externa**

Uma consulta que contém uma consulta interna ou subconsulta.

Capítulo 9

**COUNT**

Você pode contar quantas linhas são compatíveis com uma consulta SELECT sem ter que ver as linhas.

COUNT retorna um valor inteiro único.

Capítulo 6

**CREATE TABLE**

Inicia a instalação de sua tabela, mas você precisará conhecer também seus nomes de colunas e tipos de dados. Você deveria ter se preparado ao analisar o tipo de dados que colocará na sua tabela.

Capítulo 1

**CREATE TABLE AS**

Utilize este comando para criar uma tabela a partir dos resultados de um comando SELECT.

Capítulo 10

**CREATE USER**

Comando usado por alguns Sistemas SQL que permite que você crie um usuário e dê uma senha.

Capítulo 12

**CROSS JOIN (CONEXÃO CRUZADA)**

Retorna cada linha de uma tabela cruzada com cada linha de uma segunda tabela. Conhecida por vários outros nomes, incluindo Conexão Cartesiana, Conexão e não-conexão.

Capítulo 8

**D****DADOS ATÔMICOS**

Dados em suas colunas são atômicos se estão divididos até a menor parte que você precise.

Capítulo 4

**DADOS ATÔMICOS - REGRA 1**

Dados Atômicos não podem ter bits em excesso do mesmo tipo de dados na mesma coluna.

Capítulo 4

**DADOS ATÔMICOS - REGRA 2**

Dados Atômicos não podem ter colunas múltiplas com o mesmo tipo de dados.

Capítulo 4

**DELETE**

Esta é sua ferramenta para apagar linhas de dados de sua tabela. Utilize-a com sua cláusula WHERE para apontar precisamente as linhas que você quer remover.

Capítulo 3

**DISTINCT**

Retorna cada valor único apenas uma vez, sem duplicidade.

Capítulo 6

**DROP TABLE**

Permite que você apague sua tabela caso tenha cometido um erro, mas terá que fazê-lo antes de começar a utilizar o comando que permite que adicione valores para cada coluna.

Capítulo 1

**E****EQUIJOIN e NON-EQUIJOIN**

Ambas são conexões internas. O equijoin retorna linhas que são iguais, e o não-equijoin retorna qualquer linha que seja não-igual.

Capítulo 8

## Escape c\ ' e \

Escape dos apóstrofos nos seus dados de texto com um apóstrofo extra ou barra invertida na frente do apóstrofo.

Capítulo 2

## Esquema

Uma descrição dos dados em seu banco de dados bem como com qualquer outro objeto relacionado e a forma como eles se conectam.

Capítulo 7

## EXCEPT

Use esta palavra-chave para retornar apenas os valores que estão na primeira consulta, mas que NÃO ESTEJAM na segunda.

Capítulo 10

# F

## G

### GRANT

Este comando permite que você controle exatamente o que os usuários podem fazer nas tabelas e colunas baseado nos privilégios que são dados a eles.

Capítulo 12

### GROUP BY

Consolida linhas, é baseado em uma coluna comum.

Capítulo 6

# I

## INNER JOIN (CONEXÃO INTERNA)

Qualquer conexão que combina os registros de duas tabelas utilizando alguma condição.

Capítulo 8

### Inner query (consulta interna)

Uma consulta dentro de outra consulta. Também conhecida como subconsulta.

Capítulo 9

## INTERSECT

Use esta palavra-chave para retornar apenas valores que estão na primeira consulta AND (e) também na segunda consulta.

Capítulo 10

## IS NULL

Use isto para criar uma condição para testar o importuno valor NULL.

Capítulo 2

# L

LEFT OUTER JOIN (CONEXÃO EXTERNA ESQUERDA)  
UMA CONEXÃO EXTERNA ESQUERDA pega todas as linhas na tabela esquerda e combina com as linhas da tabela direita.

Capítulo 10

## LIKE C\C % and \_

Utilize LIKE com coringas para procurar por partes de linhas de texto.

Capítulo 2

## LIMIT

Permite que você especifique quantas linhas quer que sejam retornadas e em qual linha começar.

Capítulo 6

# M

## Muitos-para-muitos

Duas são conectadas por uma tabela conectora, permitindo que muitas linhas na primeira a conectar com muitas linhas na segunda, e vice-e-versa.

Capítulo 7

## MAX and MIN

Retorna o maior valor de uma coluna com MAX, e o menor com MIN.

Capítulo 6

# N

NATURAL JOIN (CONEXÃO NATURAL)

Uma conexão interna que deixa de lado a cláusula ON. Só funciona se você estiver conectando duas tabelas que tenham o mesmo nome de colunas.

Capítulo 8

# R

RIGHT OUTER JOIN (CONEXÃO EXTERNA DIREITA)

Uma CONEXÃO EXTERNA DIREITA pega todos os valores na tabela direita e combina com as linhas na tabela esquerda.

Capítulo 10

NOT (não)

NOT permite você a negar resultados e obter os valores opostos.

Capítulo 2

NULL e NOT NULL

Você também vai precisar ter uma ideia quais colunas não deveriam aceitar valores NULL para ajudar você a classificar e procurar seus dados. Você precisará definir quais colunas são NOTNULL quando você cria sua tabela.

Capítulo 1

# O

ORDER BY

Ordena alfabeticamente seus resultados baseando-se na coluna que você especificar.

Capítulo 6

# P

PRIMEIRA FORMA NORMAL (1FN)

Cada linha de dados deve conter valores atômicos e cada linha de dados deve possuir um identificador único.

Capítulo 4

# S

Segunda Forma Normal (2FN)

Sua tabela deve estar em 1FN e não conter nenhuma dependência funcional parcial para estar em 2FN.

Capítulo 7

SELECT \*

Use este comando para selecionar todas as colunas em uma tabela.

Capítulo 2

SET

Esta palavra-chave pertence a um comando UPDATE e é utilizada para alterar o valor de uma coluna existente.

Capítulo 3

SHOW CREATE TABLE

Use este comando para ver a sintaxe correta para criar uma tabela já existente.

Capítulo 4

Funções String

Permite que você modifique cópias dos conteúdos das colunas quando elas são retornadas de uma consulta. Os valores originais permanecem intocados.

Capítulo 5

Subconsulta

Uma consulta que é envolvida por outra consulta. É também conhecida como consulta interna.

Capítulo 9

## Subconsultas Não-correlacionadas

Uma subconsulta que funciona sozinha e não se refere a nada constante em uma consulta externa.

Capítulo 9

## SUM

Adiciona uma coluna de valores numéricos.

Capítulo 6

# T

## Terceira Forma Normal (3FN)

Sua tabela deve estar em 2FN e não ter dependência transitiva.

Capítulo 7

## Dependência Funcional Transitiva

Quando qualquer coluna não-chave é relacionada a qualquer outra coluna não-chave.

Capítulo 7

# U

## Um-para-muitos

Uma linha em uma tabela pode ter muitas linhas que combinam em uma segunda tabela, mas a segunda tabela só pode ter uma linha combinável na primeira.

Capítulo 7

## Um-para-um

Exatamente uma linha de uma parent table é relacionada com uma linha da tabela filha.

Capítulo 7

## UNION and UNION ALL

UNION combina os resultados de duas ou mais consultas de duas ou mais consultas em uma tabela, baseando naquilo que você especifica em uma lista de coluna do SELECT, UNION esconde os valores duplicados. UNION ALL inclui os valores duplicados.

Capítulo 10

## UPDATE

Este comando atualiza uma coluna existente ou colunas com um novo valor. Ele também usa uma cláusula WHERE.

Capítulo 3

## USE DATABASE

Leva você para dentro do banco de dados para configurar todas as tabelas.

Capítulo 1

# V

## VIEWS

Use uma view para tratar os resultados de uma consulta como se eles fossem uma tabela. Ótimo para transformar consultas complexas em simples.

Capítulo 11

## VIEWS ATUALIZÁVEIS

Estas são fotos que permitem alterar os dados nas tabelas bases. Estas views devem conter todas as linhas NULL fora da tabela base ou tabelas.

Capítulo 11

## VIEWS NÃO-ATUALIZÁVEIS

Views que não podem ser usadas para inserir (INSERT) e atualizar (UPDATE) dados na tabela base.

Capítulo 11

# W

## WITH GRANT OPTION

Permite aos usuários a darem a outros usuários os mesmos privilégios que eles têm.

Capítulo 12

# Índice Remissivo

## Números

forma normal 257  
forma normal 265  
forma normal 270

## A

CID 391  
dd 155  
dd column 155  
dd Constraint Check 374  
ALL, ANY e SOME 431  
alter table 155, 162  
AND 68, 71, 72, 81, 88  
auto\_increment 152  
avg 216

## B

between 88  
boolean 432

## C

case when else 193, 194  
CAST 434  
change column 169  
change, modify, add, drop 164, 168, 175  
chave composta 258  
chave estrangeira 245  
chave primária 146  
check 373  
conexão com subconsulta 359  
consulta e SubConsulta 314  
consulta externa e Consulta interna 317  
create database 16  
create Table com Select e Insert 286  
create Table com Select 285  
create table 18, 20, 41  
create user 405  
create View 377  
criar tabela com Union 357  
cross Join 289  
current\_User 435, 436

## D

dados atômicos 139  
DATE e Time 433  
default 43  
delete 107  
dependência funcional 261  
desc, asc 210  
distinct 219  
drop column 173  
drop table 29  
Drop View 386

## E

esquema 240  
exists e not exists 333

## F

first, after, before, last, second, third 161, 163  
Funções numéricas 436, 437

## G

Grant 406  
group by 215

## I

IN 90  
IN 312  
InnerJoin 291, 292, 294  
InnoDB e DBD 392  
insert into 31, 33, 39  
Intersect e Except 355, 356  
INT 432  
is null 83

# L

Left Outer Join 341  
left, right 178  
like 84  
limit 221

# M

max, min, count 217  
modify column 170  
muitos-para-muitos 251

# N

Natural Join 298  
not exists 332, 333  
NOT IN 89, 90

# O

OR 78, 80  
order by 204, 207

# R

rename 165  
Revoke 410  
Right outer join 347  
Role 415, 416

# S

select 39  
select 48, 51  
Select 280, 285, 286  
set password 404  
set 131  
show create table 150  
sinais '=' , '>' , '<' , '<>' , '>=' , '<=' 72, 73  
Start Transaction, commit, rollback 392, 395, 396  
Subconsulta como coluna 322  
substring, substring\_index, upper, reverse, ltrim, lenght 179  
sum 214

# T

Tipos de Conexões 301  
Tipos de dados 23  
transação 390

# U

um-para-um, um-para-muitos 250  
Union All 356  
Union 354, 355  
update 121  
Update, Substring, Length 282  
use 17

# W

where 51