# Sistemas de Gerenciamento de Banco de Dados

Tradução da
Terceira Edição

Mc Graw Hill

Ramakrishnan • Gehrke

**Sistemas de Gerenciamento de Banco de Dados**
ISBN 978-85-7726-027-0



Tradução da terceira edição do original em inglês Database Management Systems.
© 2003, 2000, 1998 de The McGraw-Hill Companies, Inc.
ISBN da obra original: 0-07-246563-8

Diretor-Geral: *Adilson Pereira*
Editora: *Gisélia Costa*
Supervisora de Produção: *Guacira Simonelli*
Preparação de Texto: *Lucrécia Freitas e Mônica de Aguiar*
Design da Capa: *Mick Wiggins*
Editoração Eletrônica: *Crontec Ltda*.

---

R165s Ramarkrishnan, Raghu.
Sistemas de gerenciamento de banco de dados [recurso eletrônico] / Raghu Ramarkrishnan,Johannes Gehrke ; tradução: Célia Taniwake. – 3. ed. – Dados eletrônicos. – Porto Alegre : AMGH, 2011.

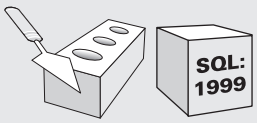
Editado também como livro impresso em 2008.
ISBN 978-85-63308-77-1

1. Ciência da computação. 2. Bases de dados – Gerenciamento. I. Gehrke, Johannes. II. Título.

CDU 004.658

---

Catalogação na publicação: Ana Paula Magnus – CRB 10/2052




# 3

# O MODELO RELACIONAL

- Como os dados são representados no modelo relacional?
- Quais restrições de integridade podem ser expressas?
- Como os dados podem ser criados e modificados?
- Como os dados podem ser manipulados e consultados?
- Como podemos criar, modificar e consultar tabelas usando SQL?
- Como obtemos um projeto de banco de dados relacional com base em um diagrama ER?
- O que são visões e por que elas são usadas?
- **Conceitos-chave:** relação, esquema, instância, tupla, campo, domínio, grau, cardinalidade; DDL SQL, `CREATE TABLE`, `INSERT`, `DELETE`, `UPDATE`; restrições de integridade, restrições de domínio, restrições de chave, `PRIMARY KEY`, `UNIQUE`, restrições de chave estrangeira, `FOREIGN KEY`; manutenção da integridade referencial, restrições adiadas e imediatas; consultas relacionais; projeto lógico de banco de dados, transformando diagramas ER em relações, expressando restrições de ER usando SQL; visões, visões e independência lógica, segurança; criando visões em SQL, atualizando visões, consultando visões, eliminando visões.

> Tabela: uma organização de palavras, números ou sinais, ou uma combinação deles, como em colunas paralelas, para exibir um conjunto de fatos ou relações em uma forma definitiva, compacta e abrangente; uma sinopse ou esquema.
>
> — *Webster's Dictionary of the English Language*

Codd propôs o modelo de dados relacional em 1970. Naquela época, a maioria dos sistemas de banco de dados era baseada em um de dois modelos de dados mais antigos (o modelo hierárquico e o modelo de rede); o modelo relacional revolucionou a área de banco de dados e suplantou em muito esses modelos iniciais. Foram desenvolvidos protótipos de sistemas de gerenciamento de banco de dados relacional em projetos de pesquisa pioneiros na IBM e na UC-Berkeley, em meados dos anos 1970, e, logo depois disso, vários fabricantes estavam oferecendo produtos de banco de dados relacional.

Atualmente, o modelo relacional é de longe o modelo de dados dominante e é a base dos SGBDs líderes do mercado, incluindo a família DB2 da IBM, o Informix, o Oracle, o Sybase, o Access e o SQLServer, da Microsoft, o FoxBase e o Paradox. Os sistemas de banco de dados relacional são onipresentes no mercado e representam um negócio de bilhões de dólares.

**SQL:** Originalmente desenvolvida como linguagem de consulta do SGBD relacional pioneiro da IBM, o System-R, a linguagem de consulta estruturada (SQL, de Structured Query Language) tornou-se a mais usada para criar, manipular e consultar SGBDs relacionais. Como muitos fabricantes oferecem produtos SQL, há necessidade de um padrão que defina a 'SQL oficial'. A existência de um padrão permite aos usuários medir a inteireza da versão de SQL de determinado fabricante. O padrão também permite aos usuários distinguir recursos da SQL específicos de um produto daqueles que são padronizados; um aplicativo que conta com recursos não padronizados é menos portável.

O primeiro padrão SQL foi desenvolvido em 1986 pelo American National Standards Institute (ANSI) e foi chamado SQL-86. Houve uma pequena revisão em 1989, chamada SQL-89, e uma revisão maior, em 1992, chamada SQL-92. A International Standards Organization (ISO) colaborou com o ANSI no desenvolvimento do padrão SQL-92. Atualmente, a maioria dos SGBDs comerciais suporta a versão SQL:1999 do padrão, uma extensão importante da SQL-92 adotada recentemente. Nossa abordagem da SQL é baseada no padrão SQL:1999, mas também é aplicável ao padrão SQL-92; os recursos exclusivos do padrão SQL:1999 são mencionados explicitamente.

O modelo relacional é muito simples e elegante: um banco de dados é uma coleção de uma ou mais *relações*, em que cada relação é uma tabela com linhas e colunas. Essa representação tabular simples permite que até usuários iniciantes entendam o conteúdo de um banco de dados e possibilita o uso de linguagens de alto nível simples para consultar os dados. As principais vantagens do modelo relacional em relação aos modelos de dados mais antigos são sua representação de dados simples e a facilidade com que mesmo consultas complexas podem ser expressas.

Embora nos concentremos nos conceitos subjacentes, também apresentaremos os recursos da **Data Definition Language** (**DDL** — linguagem de definição de dados) da SQL, a linguagem padrão para criar, manipular e consultar dados em um SGBD relacional. Isso nos permite basear a discussão firmemente em termos de sistemas de banco de dados reais.

Discutiremos o conceito de relação na Seção 3.1 e mostraremos como se faz para criar relações usando a linguagem SQL. Um componente importante de um modelo de dados é o conjunto de construtores que ele fornece para especificar as condições que devem ser satisfeitas pelos dados. Tais condições, chamadas *restrições de integridade* (RIs), permitem que o SGBD rejeite operações que poderiam corromper os dados. Apresentaremos as restrições de integridade no modelo relacional na Seção 3.2, junto com uma discussão sobre o suporte da SQL para RIs. Discutiremos como um SGBD impõe restrições de integridade na Seção 3.3.

Na Seção 3.4, examinaremos o mecanismo para acessar e recuperar dados do banco de dados, as *linguagens de consulta*, e apresentaremos os recursos de consulta da SQL, os quais examinaremos com mais detalhes em capítulo posterior.

Em seguida, discutiremos a conversão de um diagrama ER em um esquema de banco de dados relacional, na Seção 3.5. Apresentaremos as *visões* ou tabelas definidas usando consultas, na Seção 3.6. As visões podem ser usadas para definir o esquema externo de um banco de dados e, assim, fornecer o suporte para a independência lógica dos dados no modelo relacional. Na Seção 3.7, descreveremos os comandos SQL para destruir e alterar tabelas e visões.

Finalmente, na Seção 3.8, estenderemos nosso estudo de caso de projeto, a loja na Internet apresentada na Seção 2.8, mostrando como o diagrama ER de seu esquema conceitual pode ser mapeado para o modelo relacional e como o uso de visões pode ajudar nesse projeto.

## 3.1 INTRODUÇÃO AO MODELO RELACIONAL

O principal construtor para representar dados no modelo relacional é a **relação**. Uma relação consiste em um **esquema de relação** e em uma **instância de relação**. A instância de relação é uma tabela, e o esquema de relação descreve os cabeçalhos de coluna da tabela. Primeiro, descreveremos o esquema de relação e depois a instância de relação. O esquema especifica o nome da relação, o nome de cada **campo** (ou **coluna** ou **atributo**) e o **domínio** de cada campo. Um domínio é descrito em um esquema de relação pelo **nome de domínio** e tem um conjunto de **valores** associados.

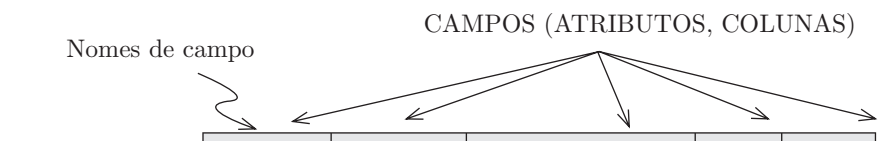
Usaremos o exemplo das informações de aluno em um banco de dados de uma universidade, do Capítulo 1, para ilustrarmos as partes de um esquema de relação:

Alunos (*id-aluno:* `string`, *nome:* `string`, *login:* `string`,
*idade:* `integer`, *média:* `real`)

Isso informa, por exemplo, que o campo chamado *id-aluno* tem um domínio denominado `string`. O conjunto de valores associados ao domínio `string` é o conjunto de todas as strings de caracteres.

Agora, veremos as instâncias de uma relação. Uma **instância** de uma relação é um conjunto de **tuplas**, também chamadas **registros**, no qual cada tupla tem o mesmo número de campos que o esquema de relação. Uma instância de relação pode ser considerada uma *tabela* na qual cada tupla é uma *linha* e todas as linhas têm o mesmo número de campos. (O termo *instância de relação* é freqüentemente abreviado apenas como *relação*, quando não há confusão com outros aspectos de uma relação, como seu esquema.)

Uma instância da relação Alunos aparece na Figura 3.1. A instância $A1$ contém seis tuplas e, conforme esperávamos, com base no esquema, ela tem cinco campos. Note que não existem duas linhas idênticas. Isso é um requisito do modelo relacional — cada relação é definida como um *conjunto* de tuplas ou linhas únicas.




| *id-aluno* | *nome* | *login* | *idade* | *média* |
|---|---|---|---|---|
| 50000 | Dave | dave@cs | 19 | 3,3 |
| 53666 | Jones | jones@cs | 18 | 3,4 |
| 53688 | Smith | smith@ee | 18 | 3,2 |
| 53650 | Smith | smith@math | 19 | 3,8 |
| 53831 | Madayan | madayan@music | 11 | 1,8 |
| 53832 | Guldu | guldu@music | 12 | 2,0 |

TUPLAS (REGISTROS, LINHAS)

**Figura 3.1** Uma instância $A1$ da relação Alunos.

Na prática, os sistemas comerciais permitem que as tabelas tenham linhas duplicadas, mas supomos que uma relação é mesmo um conjunto de tuplas, a não ser que seja mencionado de outra forma. A ordem na qual as linhas são listadas não é importante. A Figura 3.2 mostra a mesma instância de relação. Se os campos são nomeados, como em nossas definições de esquema e figuras representando instâncias de relação, a ordem dos campos também não importa. Entretanto, uma convenção alternativa é listar os campos em uma ordem específica e referir-se a um campo por sua posição. Assim, *id-aluno* é o campo 1 de Alunos, *login* é o campo 3 e assim por diante. Se essa convenção for usada, a ordem dos campos terá significado. A maioria dos sistemas de banco de dados usa uma combinação dessas convenções. Por exemplo, na SQL, a convenção dos campos nomeados é usada em instruções que recuperam tuplas, e a convenção dos campos ordenados é comumente usada ao se inserir tuplas.

| *id-aluno* | *nome* | *login* | *idade* | *média* |
|---|---|---|---|---|
| 53831 | Madayan | madayan@music | 11 | 1,8 |
| 53832 | Guldu | guldu@music | 12 | 2,0 |
| 53688 | Smith | smith@ee | 18 | 3,2 |
| 53650 | Smith | smith@math | 19 | 3,8 |
| 53666 | Jones | jones@cs | 18 | 3,4 |
| 50000 | Dave | dave@cs | 19 | 3,3 |

**Figura 3.2** Uma representação alternativa da instância $A1$ de Alunos.

Um esquema de relação especifica o domínio de cada campo ou coluna na instância de relação. Essas **restrições de domínio** no esquema especificam uma condição importante que queremos que cada instância da relação satisfaça: os valores que aparecem em uma coluna devem ser extraídos do domínio associado a essa coluna. Assim, em termos de linguagem de programação, o domínio de um campo é basicamente o *tipo* desse campo e restringe os valores que podem aparecer no campo.

Mais formalmente, seja R($f_1$:`D1`, ..., $f_n$:`Dn`) um esquema de relação, e para cada $f_i$, $1 \leq i \leq n$, seja $Dom_i$ o conjunto de valores associados ao domínio chamado `Di`. Uma instância de R que satisfaça as restrições de domínio no esquema é um conjunto de tuplas com $n$ campos:

$$\{ \langle f_1 : d_1, \ldots , f_n : d_n \rangle \mid d_1 \in Dom_1, \ldots , d_n \in Dom_n \}$$

Os sinais $\langle ... \rangle$ identificam os campos de uma tupla. Usando essa notação, a primeira tupla de Alunos mostrada na Figura 3.1 é escrita como $\langle$*id-aluno:* 50000, *nome:* Dave, *login:* dave@cd, *idade:* 19, *média:* 3,3$\rangle$. As chaves $\{...\}$ denotam um conjunto (de tuplas, nessa definição). A barra vertical | deve ser lida como 'tal que', o símbolo $\in$ deve ser lido como 'em' e a expressão à direita da barra vertical é uma condição que deve ser satisfeita pelos valores de campo de cada tupla do conjunto. Portanto, uma instância de R é definida como um conjunto de tuplas. Os campos de cada tupla devem corresponder aos campos do esquema de relação.

As restrições de domínio são tão fundamentais no modelo relacional que daqui por diante consideraremos apenas instâncias de relação que as satisfazem; portanto, *instância de relação* significa *instância de relação que satisfaz as restrições de domínio no esquema de relação.*

O **grau**, também chamado **aridade**, de uma relação é o número de campos. A **cardinalidade** de uma instância de relação é o número de tuplas que ela contém. Na Figura 3.1, o grau da relação (o número de colunas) é cinco e a cardinalidade dessa instância é seis.

Um **banco de dados relacional** é uma coleção de relações com nomes distintos. O **esquema de banco de dados relacional** é a coleção de esquemas das relações presentes no banco de dados. Por exemplo, no Capítulo 1, discutimos o banco de dados de uma universidade com relações chamadas Alunos, Professores, Cursos, Salas, Matriculado, Ministra e Aula. Uma **instância** de um banco de dados relacional é uma coleção de instâncias de relação, uma por esquema de relação no esquema de banco de dados; naturalmente, cada instância de relação deve satisfazer as restrições de domínio nesse esquema.

### 3.1.1 Criando e Modificando Relações Usando SQL

A linguagem SQL padrão usa a palavra *tabela* para denotar *relação* e freqüentemente seguimos essa convenção ao discutirmos a SQL. O subconjunto da SQL que suporta a criação, exclusão e modificação de tabelas é chamado Data Definition Language (DDL — linguagem de definição de dados). Além disso, embora exista um comando que permite aos usuários definirem novos domínios, análogo aos comandos de definição de tipo em uma linguagem de programação, deixaremos a discussão sobre definição de domínio para a Seção 5.7. Por enquanto, consideraremos apenas os domínios que são tipos internos, como `integer`.

A instrução `CREATE TABLE` é usada para definir uma nova tabela.[1] Para criarmos a relação Alunos, podemos usar a seguinte instrução:

```
CREATE TABLE Alunos (id-aluno  CHAR(20),
                     nome      CHAR(30),
                     login     CHAR(20),
                     idade     INTEGER,
                     média     REAL )
```

As tuplas são inseridas usando-se o comando `INSERT`. Podemos inserir uma única tupla na tabela Alunos, como segue:

```
INSERT
INTO     Alunos   (id-aluno, nome, login, idade, média)
VALUES   (53688, 'Smith', 'smith@ee', 18, 3,2)
```

Opcionalmente, podemos omitir a lista de nomes de coluna na cláusula `INTO` e listar os valores na ordem apropriada, mas é considerado boa prática ser explícito quanto aos nomes de coluna.

Podemos excluir tuplas usando o comando `DELETE`. Podemos excluir todas as tuplas de Alunos com *nome* igual a Smith, usando o comando:

```
DELETE
FROM     Alunos A
WHERE    A.nome = 'Smith'
```

Podemos modificar os valores de coluna em uma linha existente usando o comando `UPDATE`. Por exemplo, podemos incrementar a idade e decrementar a média do aluno com *id-aluno* 53688:

[1] A SQL também fornece instruções para destruir tabelas e alterar as colunas associadas a uma tabela; discutiremos essas instruções na Seção 3.7.

```
UPDATE  Alunos A
SET     A.idade = A.idade + 1, A.média = A.média – 1
WHERE   A.id-aluno = 53688
```

Esses exemplos ilustram alguns pontos importantes. A cláusula **WHERE** é aplicada primeiro e determina quais linhas devem ser modificadas. Então, a cláusula **SET** determina como essas linhas devem ser modificadas. Se a coluna que está sendo modificada também é usada para determinar o novo valor, o valor usado na expressão no lado direito do sinal de igualdade (=) é o valor *antigo*; ou seja, antes da modificação. Para ilustrar melhor esses pontos, considere a seguinte variação da consulta anterior:

```
UPDATE  Alunos A
SET     A.média = A.média – 0,1
WHERE   A.média >= 3,3
```

Se essa consulta for aplicada na instância $A1$ de Alunos mostrada na Figura 3.1, obteremos a instância que aparece na Figura 3.3.

| *id-aluno* | *nome* | *login* | *idade* | *média* |
|---|---|---|---|---|
| 50000 | Dave | dave@cs | 19 | 3,2 |
| 53666 | Jones | jones@cs | 18 | 3,3 |
| 53688 | Smith | smith@ee | 18 | 3,2 |
| 53650 | Smith | smith@math | 19 | 3,7 |
| 53831 | Madayan | madayan@music | 11 | 1,8 |
| 53832 | Guldu | guldu@music | 12 | 2,0 |

**Figura 3.3** Instância $A1$ de Alunos após a atualização.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.