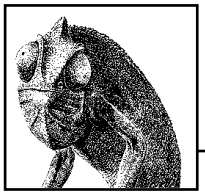

1

História e Implementações do SQL

No início dos anos 70, o trabalho produtivo do colega de pesquisa da IBM Dr. E. F. Codd levou ao desenvolvimento de um produto modelo de dado relacional chamado SEQUEL ou *Linguagem de Consulta em Inglês Estruturado (em inglês: Strutucred English Query Language)*.. SEQUEL ultimamente se transformou em SQL ou *Linguagem de Consulta Estruturada (em inglês: Structured Query Language)*.

IBM, junto com outros fornecedores de banco de dados relacionais, queria um método padronizado para acessar e manipular dados em um banco de dados relacional. Embora IBM tenha sido a primeira a desenvolver a teoria de banco de dados relacional, a Oracle foi a primeira a comercializar a tecnologia. Através do tempo, SQL se provou popular o suficiente no mercado de trabalho para atrair a atenção do American National Standards Institute (ANSI), que lançou padrões para SQL em 1986, 1989, 1992, 1999, 2003 e 2006. Este texto cobre o padrão ANSI 2003, pois o padrão 2006 lida com elementos do SQL fora do escopo dos comandos descritos neste livro. (Em essência, o padrão SQL2006 descreve como o XML deveria ser usado no SQL.)

Desde 1986, várias linguagens concorrentes permitiram programadores e desenvolvedores a acessarem e manipularem dados relacionais. No entanto, poucos foram fáceis de aprender ou aceitos universalmente como SQL. Programadores e administradores agora têm o benefício de serem capazes de aprender uma única linguagem que, com pequenos ajustes, é aplicável a uma vasta variedade de plataformas de banco de dados, aplicações e produtos.

SQL - O Guia Eessencial, Terceira Edição, fornece a sintaxe para cinco implementações comuns de SQL2003 (SQL3):

- O padrão ANSI SQL
- MySQL versão 5.1
- Oracle Database 11g
- PostgreSQL versão 8.3
- SQL Server 2008 da Microsoft

O Modelo Relacional e ANSI SQL

Sistemas gerenciadores de banco de dados relacionais (SGBDRs) como aqueles apresentados neste livro são os motores principais de sistemas de informação mundial e particularmente de aplicações na web e sistemas computacionais distribuídos cliente/servidor. Eles capacitam uma multidão de usuários para acessar, criar, editar e manipular dados rapidamente e simultaneamente sem impactar outros usuários. Também permitem desenvolvedores escreverem aplicações úteis para acessar seus recursos e fornecem aos administradores as habilidades que eles precisam para manter, assegurar e aperfeiçoar recursos de dados organizacionais.

Um SGBDR é definido como um sistema que usuários observam dados como uma coleção de tabelas relacionadas entre si através de valores de dados comuns. Dados são armazenados em *tabelas*, que são compostas por *linhas* e *colunas*. Tabelas de dados independentes podem ser unidas (ou relacionadas) umas as outras, se cada uma delas tiver colunas de dados únicas e identificadas (chamadas *chaves*) que representam valores de dados tidos em comum. E. F. Codd descreveu primeiro a teoria de banco de dados relacional em seu ponto de referência "Um Modelo Relacional de Dados para Grandes Bancos de Dados Compartilhados", publicado em *Comunicações da ACM* (Association for Computing Machinery) em Junho, 1970. Sob o novo modelo de dados relacionais de Codd, dados foram *estruturados* (em tabelas de linhas e colunas); *manipuláveis* usando operações como seleções, projeções, e junções e *consistentes* como resultado de regras de integridade como chaves e integridade referencial. Codd também articulou regras que governaram como os bancos de dados relacionais deveriam ser criados. O processo para aplicar estas regras é agora conhecido como *normalização*.

Regras de Codd para Sistemas de Banco de Dados Relacional

Codd aplicou teorias matemáticas rigorosas (primeiramente teoria de conjunto) na administração de dados, e ele obedeceu a uma lista de critérios que um banco de dados deve encontrar para ser considerada relacional. Em sua parte mais importante, os núcleos (em inglês: core) de conceitos de base de dados relacionais sobre armazenar dados em tabelas. Este conceito é agora tão comum que parece trivial, no entanto, não muito tempo atrás o objetivo de criar um sistema capaz de sustentar o modelo relacional foi considerado um tiro distante com utilidade limitada.

A seguir estão os *Doze Princípios de Banco de Dados Relacionais de Codd*:

1. Informação é representada logicamente em tabelas.
2. Dados devem ser logicamente acessíveis por tabelas, chaves primárias e colunas.
3. Valores NULLs devem ser tratados uniformemente como "informação perdida", não como strings vazias, espaços ou zeros.
4. Metadados (dados sobre o banco de dados) devem ser armazenados no banco de dados assim como são os dados regulares.
5. Uma única linguagem deve ser capaz de definir dados, exibições ou visualizações (em inglês: view), restrições de integridade, autorização, transações e manipulação de dados.
6. Exibições ou visualizações (em inglês: view) devem mostrar as atualizações de suas tabelas de base e vice versa.
7. Uma única operação deve ser disponível para fazer cada uma das operações seguintes: recuperar, inserir, atualizar ou remover dados.
8. Lotes (em inglês: Batch) e operações do usuário final são logicamente separados do armazenamento físico e métodos de acesso.
9. Lotes (em inglês: Batch) e operações do usuário final podem mudar o esquema do banco de dados sem ter que recriá-lo ou as aplicações construídas por meio dele.

10. Restrições de integridade devem ser disponíveis e armazenados em metadados, não em um programa de aplicação.
11. A linguagem de manipulação de dados do sistema relacional não deveria se importar onde ou como dados físicos são distribuídos e não deveria requerer alteração se dados físicos são centralizados ou distribuídos.
12. Qualquer processamento de linha feito no sistema deve obedecer às mesmas regras de integridade e restrições que os grupos de operações de processamento obedecem.

Estes princípios continuam sendo decisões usadas para validar as características "relacionais" de uma plataforma de banco de dados; um banco de dados que não encontra todas estas regras não é totalmente relacional. Enquanto estas regras não se aplicam ao desenvolvimento de aplicações, elas determinam se o próprio motor (em inglês: engine) do banco de dados pode ser considerado realmente "relacional". Recentemente, a maiorias dos produtos comerciais SGBDR passam no teste de Codd. Entre as plataformas discutidas em *SQL O Guia Essencial*, Terceira Edição, somente MySQL fracassou em suportar todos estes requisitos, e somente então em lançamentos prévios ao cobertos neste livro.

Entender os princípios de Codd ajuda programadores e desenvolvedores no desenvolvimento adequado e no projeto de bancos de dados relacionais (RDBs). As seções seguintes detalham como alguns destes requisitos são encontrados em SQL usando RDBs.

Estruturas de dados (Regras 1,2, e 8)

As regras 1 e 2 de Codd relatam que "informação é representada logicamente em tabelas" e que "dados devem ser logicamente acessíveis por tabela, chaves primárias e colunas." Assim, o processo de definir uma tabela para um banco de dados relacional não exige que os programas instruam o banco de dados em como interagir com estruturas de dados físicos. Além disso, SQL logicamente isola os processos de acessar dados e manter fisicamente aquele dado, como requerido pela regra 8: "lote (em inglês: batch) e operações do usuário final são logicamente separados de armazenamento físico e métodos de acesso."

No modelo relacional, dados são mostrados logicamente como uma *tabela* em duas dimensões que descreve uma única entidade (por exemplo, gastos de negócios). Acadêmicos se referem às tabelas como *entidades* e às colunas como *atributos*. Tabelas são compostas por *linhas*, ou *registros* (acadêmicos os chamam de *tuplas*) e *colunas* (chamadas de *atributos*, pois cada coluna de uma tabela descreve um atributo específico da entidade). A intersecção de um registro e de uma coluna fornece um único *valor*. A coluna ou colunas de quais valores identificam unicamente cada registro podem agir como uma *chave primária*. Ultimamente esta representação parece elementar, mas era na verdade um tanto inovadora quando foi proposta primeiramente.

SQL3 define toda uma hierarquia de estrutura de dados além de tabelas simples, embora tabelas sejam as estruturas de dados mais importantes. O padrão relacional lida com dados em uma base de tabela por tabela, não em uma base de registro por registro. Esta orientação de tabela central é o coração do conjunto de programação. Consequentemente, quase todos os comandos SQL operam com muito mais eficiência contra conjuntos de dados dentro ou através de tabelas do que contra registros individuais. Dito de outro jeito, uma programação de SQL efetiva requer que você pense em termos de conjuntos de dados, do que em linhas individuais.

Figura 1-1 é uma descrição da terminologia SQL3 usada para descrever a estrutura de dados hierárquica usada por um banco de dados relacional: *clusters* contêm conjuntos de *catálogos*; *catálogos* contêm conjuntos de *esquemas*; *esquemas* contêm conjuntos de *objetos*, como *tabelas e exibições ou visualizações (em inglês: view)*; e *tabelas* são compostas por conjuntos de *colunas* e *registros*.

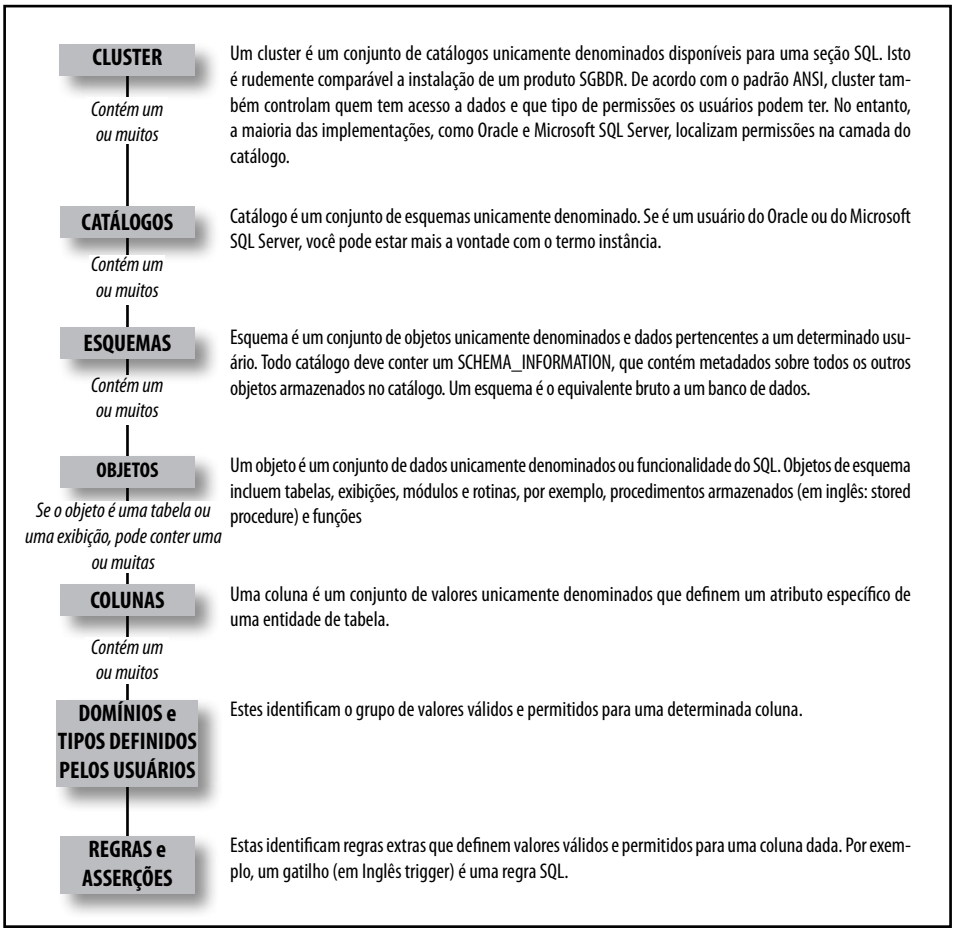

Figura 1-1. Hierarquia do conjunto de dados do SQL3

Por exemplo, em uma tabela **Business_Expense**, uma coluna chamada **Expense_Date** pode mostrar quando uma despesa foi incluída. Cada registro na tabela descreve uma entidade específica, neste caso, tudo que cria uma despesa de negócio (quando aconteceu; quanto custou; quem fez a despesa; para que foi e assim por diante).

Cada atributo de uma despesa – em outras palavras, cada coluna – deve ser *atômico*, isto é, cada coluna deve conter um, e somente um, valor. Se uma tabela é construída de modo que a intersecção de uma linha e uma coluna pode conter mais de um valor distinto, um dos padrões primitivos de design do SQL foi violado. (Algumas plataformas de banco de dados discutidas neste livro permitem você a colocar mais de um valor em uma coluna, através dos tipos de dados VARRAY ou TABLE).

Regras de comportamento são especificadas para valores de colunas. Para a maioria, valores de coluna devem compartilhar um *domínio* comum, mais conhecido como um *tipo de dado*. Por exemplo, se o campo **Expense_date** é definido como tendo um tipo de dado de DATE, o valor ELMER não deve ser substituído por aquele campo porque é um texto, não uma data e o campo **Expense_date** só pode conter datas. Além disso, o SQL3 permite mais controle de valores de colunas através da aplicação de *restrições (em inglês: constraints)* (discutido em detalhe no capítulo 2) e *e asserções (em inglês: assertions)*. Uma restrição do SQL3 deve, por exemplo, limitar **Expense_date** em despesas com menos de um ano. Também, acessos de dados para todos os indivíduos e processos computacionais são controlados no nível de esquema por um *Identificador de Autorização (em inglês: AuthorizationID)* ou *usuário*. Permissões para acessar ou modificar conjuntos específicos de dados podem ser concedidos ou restritos em uma base por usuário.

Bancos de dados SQL também empregam *conjuntos de caracteres* e *comparações*. Conjuntos de caracteres são os "símbolos" ou "alfabetos" usados pela "linguagem"do dado. Por exemplo, o conjunto de caracteres em inglês americano não contém o caractere especial para ñ no conjunto de caractere espanhol. Comparações são conjuntos de certas regras que operam em um conjunto de caracteres. Uma comparação define como uma operação de manipulação de determinados dados os classificam. Por exemplo, um conjunto de caracteres em inglês americano pode ser classificado tanto como *ordem de caractere, sensível ou não ao tamanho da letra (em inglês: character-order, case-insensitive, ou case-sensitive)*.

O padrão ANSI não diz como classificações devem ser feitas, somente plataformas devem fornecer comparações comuns encontradas em uma língua particular.

É importante saber qual comparação você está usando quando escreve um código SQL contra uma plataforma de banco de dados, pois pode ter um impacto direto em como consultas se comportam, e particularmente no comportamento de cláusulas de WHERE e ORDER BY de instruções SELECT. Por exemplo, uma consulta que classifica dados usando uma comparação binária retornará dados em uma ordem bem diferente daquela que classifica dados usando, digamos, uma comparação em inglês americano.

Nulo (em inglês: Null) (Regra 3)

A maioria dos bancos de dados permite que qualquer um de seus tipos de dados suportados armazenem valores NULLs. Programadores e desenvolvedores de SQL inexperientes tendem a pensar em NULL como zero ou espaços em branco. De fato, NULL não é nenhum destes. Em SQL3, NULL literalmente significa que o valor é desconhecido ou indeterminado (Esta questão sozinha – se NULL deve ser considerado desconhecido ou indeterminado – é o assunto de muitos debates acadêmicos). Esta diferenciação capacita um projetista de banco de dados distinguir entre estas entradas que representam um zero deliberadamente colocado, por exemplo, e aquelas onde cada dado não é gravado no sistema ou um NULL entrou explicitamente. Como uma ilustração desta diferença semântica, considere um sistema que localiza pagamentos. Se um produto

tem um preço NULL que não significa que o produto é grátis; pelo contrário, um preço NULL indica que a quantidade é desconhecida ou talvez não foi ainda determinada.

Há uma grande quantidade de diferenciação entre a plataforma de banco de dados em termos de como elas lidam com valores NULLs. Isto leva a alguns grandes problemas de conexão entre aquelas plataformas relacionadas com NULLs. Por exemplo, uma string vazia (em outras palavras, uma string NULL) é inserida como uma valor NULL no Oracle. Todas as outras bases de dados cobertas neste livro permitem a inserção de uma string vazia em colunas VARCHAR e CHAR.

Um efeito colateral de natureza indeterminada de um valor NULL é que não pode ser usado em um cálculo ou comparação. Aqui estão algumas regras breves mais muito importantes, de um padrão ANSI, para lembrar sobre o comportamento de valores NULLs quando lidar com NULLs em instruções do SQL:

- Um valor NULL não pode ser inserido em uma coluna definido como NOT NULL.
- Valores NULLs não são iguais aos outros. Este é um erro frequente para comparar duas colunas que contêm NULL e esperam que valores NULLs combinem. (O jeito apropriado de identificar um valor NULL é pela cláusula WHERE ou em uma expressão booleana usar frases como "o valor IS NULL"e "o valor IS NOT NULL")
- Uma coluna contendo um valor NULL é ignorada no cálculo de valores agregados como AVG, SUM, ou COUNT MAX.
- Quando colunas que contêm valores NULL são listadas na cláusula GROUP BY de uma consulta, a consulta emissora contém uma única linha de valores NULL. Em essência, o padrão ANSI considera todos os NULLs em um único grupo.
- Cláusulas DISTINCT e ORDER BY, como GROUP BY, também vêem valores NULLs como indistinguíveis uns dos outros. Com a cláusula ORDER BY, o fornecedor é livre para escolher se os valores NULL são ordenados acima (por primeiro, ou no inicio, no conjunto de resultados) ou abaixo (último no conjunto de resultados) por padrão.

Metadados (regras 4 e 10)

A quarta regra de Codd para os bancos de dados relacionais declara que dados sobre o banco de dados devem ser armazenados em tabelas padrões, assim como todos os outros dados são. Dados que descrevem o próprio banco de dados são chamados *metadados*. Por exemplo, toda vez que você criar uma nova tabela ou ou uma exibição (em inglês: view) em um banco de dados, registros são criados e armazenados que descrevem a nova tabela. Registros adicionais são necessários para armazenar quaisquer colunas, chaves ou restrições na tabela. Esta técnica é implementada na maioria dos bancos de dados fonte aberta (em inglês: open source) e comerciais. Por exemplo, o SQL Server usa as chamadas "tabelas de sistemas" (em inglês: system tables) para localizar toda a informação sobre o banco de dados, tabelas e objetos de banco de dados, em qualquer banco de dados. Também tem "sistemas de banco de dados" (em inglês: system databases) que rastream a informação sobre o servidor em qual o banco de dados é instalado e configurado.

A linguagem (regras 5 e 11)

As regras de Codd não requerem que o SQL seja usado com um banco de dados relacional. Suas regras, particularmente as regras 5 e 11, somente especificam como a linguagem deve se comportar quando acoplada com um banco de dados relacional. Em certo momento SQL competiu com outras linguagens (como RDO da Digital e Fox/PRO) que podem ter encaixado na conta relacional, mas o SQL venceu, por três razões. Primeiro, o SQL é uma linguagem relativamente simples, intuitiva, assim como uma língua como o inglês que lida com a maioria dos aspectos de manipulação de dados. Segundo, o SQL é de um satisfatório alto nível. Um programador ou administrador de banco de dados (DBA) não tem que perder tempo assegurando que dados são armazenados no registro apropriado de memória ou que dados são ocultados no disco; o sistema de administração de banco de dados (DBMS) lida com esta tarefa automaticamente. Finalmente, devido a nenhum fornecedor sozinho possuir um SQL, ele foi adotado através de várias plataformas.

Exibições ou Visualizações (em inglês: View)

Uma exibição ou visualização (em inglês: view) é uma tabela virtual que não existe como um repositório físico de dados, mas é pelo contrário construído simultaneamente a partir de uma instrução SELECT sempre que a exibição é executada ou consultada (em inglês: queried). Exibições ou visualizações (em inglês: view) capacitam você a construir diferentes representações da mesma fonte de dados para uma variedade de públicos sem ter que alternar a forma em que os dados são armazenados.

Alguns fornecedores suportam objetos de banco de dados chamados *Exibições materializadas (em inglês: materialized views)*. Não deixe a similiaridade dos itens confundir você; *que exibições materializadas (em inglês: materialized views)* não são governadas pelas mesmas regras que views do padrão ANSI.

Operações em conjuntos (regras 7 e 12)

Outras linguagens de manipulação de dados, assim como a venerável Xbase, realizam suas operações de dados bem diferentemente do SQL. Estas linguagens requerem que você diga ao programa exatamente como tratar o dado, um registro por vez. Como os programas circulam através de uma lista de registros, realizando sua lógica, um registro de cada vez, este estilo de programação é frequentemente chamado de *processamento de linha* ou *programação procedural.*

Em contraste, os programas do SQL operam em conjuntos de dados lógicos. A teoria de conjunto é aplicada em quase todas as instruções SQL, incluindo instruções de SELECT, INSERT, UPDATE e DELETE. Em efeito, dados são selecionados por um conjunto chamado de "tabela". Diferente do estilo de processamento de linhas, *processamento de conjunto* permite os programadores a dizerem ao banco de dados simplesmente *o que* é requerido, e não *como* cada pedaço individual de dado deve ser lidado. Algumas vezes processamento de conjunto é referido como um *processamento declarativo*, desde que um programador declare somente qual dado é desejado (como em, "Me dê o nome de todos os empregados na região sudeste que ganham mais que $70,000 por ano") ao invés de descrever o procedimento exato a ser usado para recuperar ou manipular dados.

Teoria de conjunto foi a ideia original do matemático Georg Cantor, que a desenvolveu no fim do século dezenove. Naquela época, a teoria de conjunto (e a teoria de Cantor sobe o infinito) foi bem controversa. Hoje, a teoria de grupo é uma parte tão comum da vida que é aprendida no ensino fundamental. Coisas como catálogos de cartões, o sistema decimal de Dewey e as listas telefônicas em ordem alfabética são exemplos simples e comuns de teoria de conjunto aplicada.

Exemplos de teoria de conjunto juntamente com bancos de dados relacionais estão detalhados na seção a seguir.

Regras de Codd em ação: Exemplos simples de SELECT

Até este ponto, este capítulo focou nos aspectos individuais de uma plataforma de dados relacional assim como definida por Codd e implementada através do SQL ANSI. Esta seção apresenta uma visão de alto nível de instruções SQL mais importantes, SELECT, e alguns dos seus pontos salientes – especialmente, as operações relacionais conhecidas como *projeções (projections), seleções (selections), e junções (joins)*:

Projeção (Projection)
: Recupera colunas de dados específicas

Seleção (Selection)
: Recupera linhas de dados específicas

Junção (Join)
: Retorna colunas e linhas de duas ou mais tabelas em um único conjunto de resultados

Embora a primeira vista isso pareça como se a instrução SELECT lidasse somente com operações de seleção relacional, na realidade, SELECT lida com todas as três operações.

A instrução a seguir incorpora a operação de projeção por selecionar os primeiros e últimos nomes de um autor, mais seu estado natal, pela tabela de **autores**:

```
SELECT      au_prname, au_ulname, state
FROM        authors
```

Os resultados de qualquer instrução SELECT são apresentados como outra tabela de dados:

```
au_prname      au_ulname                      state
---------------------------------------------------
Johnson        White                          CA
Marjorie       Green                          CA
Cheryl         Carson                         CA
Michael        O'Leary                        CA
Meander        Smith                          KS
Morningstar    Greene                         TN
Reginald       Blotchet-Halls                 OR
Innes          del Castillo                   MI
```

Os dados resultantes são às vezes chamados de *conjunto de resultado (em inglês: result set), tabela de trabalho (em inglês: work table) ou tabela derivada (em inglês: derived table)*, diferenciando-o de uma *tabela base (em inglês: base table)* no banco de dados que é o alvo da instrução SELECT.

É importante notar que a operação relacional de projeção, não de seleção, é especificada usando a cláusula SELECT (que é a palavra-chave SELECT seguida por uma lista de expressões a serem recuperadas) de uma instrução SELECT. Seleção – a operação de recuperar linhas específicas de dados – é determinada usando a cláusula WHERE em uma instrução SELECT. WHERE filtra as linhas de dados indesejadas recuperando somente as linhas requeridas. Continuando com o exemplo anterior, a instrução a seguir selecionam autores de estados diferentes da Califórnia:

```
SELECT      au_prname, au_ulname, state
FROM        authors
WHERE       state <> 'CA'
```

Enquanto a primeira consulta recupera todos os autores, o resultado desta segunda consulta é um subconjunto de registros muito menor:

au_prname	au_ulname	state
Meander	Smith	KS
Morningstar	Greene	TN
Reginald	Blotchet-Halls	OR
Innes	del Castillo	MI

Combinando as capacidades de projeção e seleção em uma única consulta, você pode usar o SQL para recuperar somente colunas e registros que você precisar em qualquer momento.

Junções são as operações relacionais seguintes e as últimas que iremos falar nesta seção. Uma junção relaciona uma tabela à outra para devolver um conjunto de resultados feito de dados relacionais das duas tabelas.

Diferentes fornecedores permitem você a juntar números variáveis de tabela em uma única operação de junção. Por exemplo, o Oracle não estabelece limite no número de tabelas em uma junção, enquanto o Microsoft SQL Server permite até 256 em uma operação de junção.

O método padrão ANSI de junções realizadas é usar a cláusula JOIN em uma instrução SELECT. Em um método mais antigo, conhecido como uma *junção theta (em inglês: theta join)*, realiza a análise de junção na cláusula WHERE. O exemplo a seguir mostra ambas as abordagens. Cada instrução recupera informações de empregados da tabela de base de **empregados** assim como descrições de emprego da tabela de base de **empregos**. O primeiro SELECT usa a mais nova cláusula do JOIN ANSI, enquanto o segundo SELECT usa uma junção tetha (em inglês: theta join):

```
-- ANSI style
SELECT      a.au_fname, a.au_lname, t.title_id
FROM        authors AS a
JOIN        titleauthor AS t ON a.au_id = t.au_id
WHERE       a.state <> 'CA'

-- Theta style
SELECT      a.au_fname, a.au_lname, t.title_id
FROM        authors AS a,
            titleauthor AS t
WHERE       a.au_id = t.au_id
   AND      a.state <> 'CA'
```

Para mais informações sobre junções, veja a seção de "Subcláusula JOIN" no Capítulo 3.

História do Padrão SQL

Em resposta a proliferação dos dialetos do SQL, ANSI publicou seu primeiro padrão SQL em 1986 para trazer maior conformidade entre fornecedores. Isto foi seguido por um segundo padrão, adotado amplamente em 1989. A International Standards Organization (ISO) também aprovou o padrão SQL. ANSI lançou uma atualização em 1992, conhecida como SQL92 ou SQL2 e outra em 1999, chamada de SQL99 ou SQL3. A próxima atualização feita em 2003, também é referida como SQL3 (ou SQL2003). Quando nós usamos este termo neste livro, estamos nos referindo à revisão 2003 do padrão.

Cada vez que ele revisa o padrão SQL ANSI, o ANSI adiciona novos atributos e incorpora novos comandos e capacidades na linguagem. Por exemplo, o padrão SQL99 adicionou um grupo de capacidades de extensões de tipos de dados que manipulam orientação a objeto.

O Que Há de Novo no SQL2006

O corpo de padrão ANSI que regulariza o SQL lançou um novo padrão em 2006, em que as maiores melhorias do SQL3 foram retidas e aumentadas. O lançamento do ANSI SQL2006 foi evoluído em relação ao lançamento de SQL3, mas não incluiu qualquer mudança significativa para os comandos e funções do SQL3 que são descritos na segunda edição deste livro. Ao invés disso, SQL2006 descreveu uma nova área totalmente funcional de comportamento para o padrão SQL. Brevemente, o SQL2006 descreve como SQL e XML (eXtensible Markup Language) interagem. Por exemplo, o padrão SQL descreve como importar e armazenar dados XML em um banco de dados do SQL, manipular os dados, e então, publicar os dados em formas nativas de XML e como dados SQL convencionais empacotados na forma XML. O padrão SQL2006 fornece um meio de integrar códigos de aplicação SQL com XQuery, a XML Query Language padronizada pelo World Wide Web Consortium (W3C). Porque XML e XQuery são disciplinas em seus direitos, elas são consideradas além do escopo deste livro e não estão cobertas aqui.

O Que Há de Novo no SQL2003 (SQL3)

O SQL99 tinha duas partes principais, Fundamental (em inglês: Foundation): 1999 e Vínculos (em inglês: Binding): 1999. A Seção Funtamental do SQL3 inclui todos os padrões fundamentais e de vínculo do SQL99, assim como uma nova seção chamada esquemata (em inglês: schemata).

Os requisitos mais importantes do SQL3 não mudaram do núcleo do SQL99, então as plataformas de banco de dados que se adequaram ao SQL99 se adequaram automaticamente ao SQL3. Embora o Núcleo (em inglês: Core) do SQL3 não tenha adições (exceto por algumas palavras reservadas recentemente), várias instruções individuais e comportamentos têm sido atualizados ou modificados. Pelo fato destas atualizações serem refletidas nas descrições de sintaxe individuais de cada instrução no Capitulo 3, não perderemos tempo com elas aqui.

Alguns elementos do núcleo do SQL99 foram deletados no SQL3, incluindo:

- Os tipos de dados *BIT E BIT VARYING*
- A cláusula *UNION JOIN*
- A instrução *UPDATE... SET ROW*

Vários outros atributos, a maioria que foram ou são um pouco obscuros, também foram adicionados, deletados ou renomeados. Muitos destes novos atributos do padrão SQL3 são atualmente interessantes principalmente de um ponto de vista acadêmico, pois nenhuma das plataformas de banco de dados suportam eles ainda. No entanto, alguns novos atributos se tornam interessantes:

Funções OLAP elementares
: O SQL3 adiciona uma emenda de Online Analytical Processing (OLAP), incluindo várias funções de exibição parcial para suportar cálculos amplamente usados como médias de mudança e somas acumulativas. Funções de exibição parcial são agregadas, computadas através de uma janela de dados: ROW_NUMBER, RANK, DENSE_RANK, PERCENT_RANK e CUME_DIST. Funções OLAP são totalmente descritas no T611 do padrão. Algumas plataformas de banco de dados estão começando a suportar as funções OLAP. Consulte o Capitulo 4 para detalhes.

Amostragem
: O SQL3 adiciona a cláusula TABLESAMPLE à cláusula FROM. Isto é útil para consultas de estatística em bancos de dados grandes, como um data warehouse.

Funções numéricas aprimoradas
: O SQL3 adiciona um grande número de funções numéricas. Neste caso, o padrão, pois principalmente alcançando a tendência na indústria, pelo fato de que uma ou mais plataformas de banco de dados já suportavam as funções novas. Consulte o Capitulo 4 para mais detalhes.

Níveis de Conformidade

O SQL99 foi construído sobre níveis de conformidade do SQL92. O SQL92 primeiro introduziu níveis de conformidade definindo três categorias: Principiante, Intermediário e Completo (em inglês: Entry, Intermediate e Full). Fornecedores tiveram que alcançar pelo menos adaptações do nível de Entrada para reivindicar a adaptação ANSI SQL. O U.S. National Institute of Standards and Technology (NIST) adicionaram mais tarde o nível transacional entre os níveis de Entrada e Intermediário, então os níveis de adaptação NIST eram de Entrada, Transacional, Intermediário e Completo, enquanto os ANSI eram somente de Entrada, Intermediário e Completos. Cada nível mais alto do padrão foi um super conjunto do nível subordinado, significando que cada nível mais alto incluiu todos os atributos dos níveis mais baixos de conformidade.

Mais tarde, o SQL99 alterou a base de níveis de adaptação, se livrando dos níveis de Entrada, Intermediário e Completo. Com o SQL99, fornecedores devem implementar todos os atributos dos níveis mais baixos de adaptação, A Parte mais importante do SQL99, com o intuito de reivindicar (e publicar) que eles estão prontos para SQL99. A Parte mais importante do SQL99 inclui o grupo de atributos da antiga entrada do SQL92, atributos de outros níveis de SQL92, e alguns atributos novos. Um fornecedor pode também escolher implementar pacotes de atributos adicionais descritos no padrão SQL99.

Pacotes de Características Suplementares no Padrão SQL3

O padrão SQL3 representa o ideal, mas poucos fornecedores frequentemente encontram ou excedem os requisitos do núcleo (em inglês: core)do SQL3. O padrão do núcleo (em inglês: core) é como o limite de velocidade interestadual: alguns motoristas vão além e outros vão abaixo, mas poucos vão exatamente no limite de velocidade. Similarmente, implementações de fornecedores podem variar muito.

Dois comitês – um sobre ANSI, o outro sobre ISO, e ambos compostos de representantes de cada fornecedor SGBDR virtualmente – rascunhou as definições de atributos suplementares descritas nesta seção. Neste ambiente colaborativo e um tanto político, fornecedores se comprometeram em quais atributos exatos propostos e implementações poderiam ser incorporados no novo padrão.

Novos atributos no padrão ANSI frequentemente são derivados de um produto existente ou são o crescimento da nova pesquisa e desenvolvimento na comunidade acadêmica. Consequentemente, adoção de fornecedores de padrões ANSI específicos podem ser instáveis. A nova adição relativa do padrão SQL é SQL/XML (largamente expandido no SQL2006). As outras partes do padrão SQL99 continuam no SQL3, embora seus nomes podem ter sido trocados e eles podem ter sido levemente reorganizados.

Os nove pacotes de atributos suplementares, representando diferentes subconjuntos de comandos, são de implementação opcional. Alguns atributos podem aparecer em múltiplos pacotes, enquanto outros não aparecem em nenhum outro pacote. Estes pacotes e seus atributos são descritos na lista seguinte:

Parte 1, SQL*/Framework*
: Inclui definições comuns e conceitos usados através do padrão. Define a forma que o padrão é estruturado e como as várias partes se relacionam umas com as outras, e descreve os requerimentos de adaptação determinados pelo comitê do padrão.

Parte 2, SQL*/Fundamental (em inglês: Foundation)*
: Inclui o núcleo, uma argumentação da parte mais importante do SQL99. Isto é a maior e mais importante do padrão.

Parte3, SQL/CLI *(Interface de Nível de Chamada) em inglês, Call-Level Interface*
: Define a interface do nível de chamada para instruções SQL invocada dinamicamente de um programa de aplicação externa. Também inclui mais de 60 especificações de rotina para facilitar o desenvolvimento de um software protegido e realmente portável.

P*arte 4,* SQL/PSM *(Módulos armazenados de persistência) em inglês, Persistent Stored Modules*
: Padroniza a ideia de linguagem procedural similar aquelas encontradas em dialetos SQL específicos para plataformas de banco de dados como PL/SQL e Transact-SQL.

Parte 9, SQL/MED *(Gerenciamento de Dados Externos) em inglês, Management of External Data*
: Define o gerenciamento de dados localizados fora do banco de dados (atual) usando ligação de dados (em inglês: datalink) e interfaces wrapper.

Parte 10, SQL/OBJ *(Vínculo de Linguagem de Objetos) em inglês, Object Language Binding*
: Descreve como embutir instruções SQL em programas Java. É extremamente relacionado com JDBC, mas oferece algumas vantagens. Também é muito diferente da linguagem vinculada de um hospedeiro (em inglês: host) tradicional disponível em versões iniciais do padrão.

Parte 11, SQL*/Esquemata (em inglês: Schemata)*
: Define mais de 85 exibições (em inglês: views) (três a mais que o SQL99) usadas para descrever os metadados de cada banco de dados e armazenados em um esquema especial chamado *INFORMATION_SCHEMA*. Atualiza várias views que existiam no SQL99.

Parte 12, SQL/JRT *(Rotinas e Tipos Java) em inglês, Java Routine and Types*
Define várias rotinas e tipos SQL usando a linguagem de programação Java. Várias características do Java, como classes e métodos estáticos, são suportados agora.

Parte 14, XML/SQL
Adiciona um novo tipo chamado *XML*, quatro novas operações (*XMLPARSE, XMLSERIALIZE, XMLROOT e XMLCONCAT*), várias novas funções (descritas no Capítulo 4) e o novo predicado *IS DOCUMENT*. Também inclui regras para mapeamento de elementos relacionados à SQL (como **identificadores**, **esquemas** e **objetos**) para elementos relacionados à XML.

Note que as partes 5,6,7 e 8 não existem pelo projeto.

Esteja consciente que uma plataforma SGBDR pode reivindicar obediência ao SQL3 ao encontrar padrões do núcleo do SQL99, então leia as especificações do fornecedor para uma descrição completa do seu atributo de conformidade do ANSI. Entendendo o que atributos cobrem os nove pacotes, o usuário pode ter uma clara ideia das capacidades de um SGBDR particular e de como os vários atributos se comportam quando um código SQL é transportado para outros produtos de banco de dados.

Os padrões ANSI – que cobrem recuperação, manipulação e gerenciamento de dados em comandos como *SELECT, JOIN, ALTER TABLE e DROP*– formalizam muitos comportamentos SQL e estruturas de sintaxe pela variedade de plataformas. Estes padrões se tornaram ainda mais importantes que produtos de banco de dados de fonte aberta, como MySQL e PostgreSQL, cresceram em popularidade e começaram a ser desenvolvidos por equipes virtuais que por grandes corporações.

SQL O Guia Essencial, Terceira Edição, explica a implementação SQL de quatro populares SGBDR. Estes fornecedores não atendem todos os padrões SQL3; aliás, todas as plataformas SGBDR mantém contato constante com os membros responsáveis pelo padrão. Frequentemente, assim que os fornecedores fecham o padrão, os corpos de padrões atualizam, refinam ou de outra forma alteram o benchmark. Mutuamente, os fornecedores frequentemente implementam novos atributos que não são ainda uma parte do padrão mas que incentivam a efetividade dos seus usuários.

Classes de Instruções SQL

Comparando classes seguintes de instruções delineia SQL do SQL92. No entanto, os termos antigos ainda são usados, então leitores precisam conhecê-los. SQL92 agrupou instruções em três amplas categorias:

Linguagem de manipulação de dados (DML), em inglês, Data Manipulation Language
Fornece comandos de manipulação de dados específicos como *SELECT*, *INSERT*, *UPDATE* e *DELETE*.

Linguagem de definição de dados (DDL), em inglês, Data Definition Language
Contém comandos que lidam com a acessibilidade e manipulação de objetos de anco de dados, incluindo *CREATE* e *DROP*

Linguagem de Controle de dados (DCL), em inglês, Data Control Language
Contém os comandos relacionados com permissões *GRANT* e *REVOKE*

Em contraste, SQL3 fornece sete categorias mais importantes, agora chamadas de classes, que fornecem um framework geral para os tipos de comandos disponíveis no SQL. Estas instruções tentam identificar as instruções em cada classe mais exatamente e logicamente, e eles fornecem para o desenvolvimento de novos atributos e classes de instruções. Além disso, as novas classes de instruções agora permitem algumas instruções "órfãs" que não se encaixam bem em quaisquer das categorias antigas a serem devidamente classificadas.

A tabela 1-1 identifica as cláusulas de instruções SQL e lista alguns dos comandos em cada classe, cada a qual é completamente discutida mais tarde. Neste ponto, a chave é lembrar os títulos de classes de instruções.

Tabela 1-1. Classes de instruções SQL3

Classe	Descrição	Exemplos de comandos
Instruções de conexão SQL	Inicializa e finaliza uma conexão de cliente	*CONNECT, DISCONNECT*
Instruções de controle SQL	Controla a execução de um conjunto de instruções SQL	*CALL, RETURN*
Instruções de dados SQL	Pode ter um efeito persistente e duradouro nos dados	*SELECT, INSERT, UPDATE, DELETE*
Instruções de diagnóstico SQL	Fornece informação de diagnóstico e aumenta exceções e erros	*GET DIAGNOSTICS*
Instruções de esquema SQL	Pode ter um efeito persistente e duradouro no esquema de banco de dados nos objetos em cada esquema	*ALTER, CREATE, DROP*
Instruções de seção SQL	Controla comportamento defeituoso e outros parâmetros para uma seção	*SET instruções, como SET CONSTRAINT*
Instruções de transação SQL	Define o ponto inicial e final de uma transação	*COMMIT, ROLLBACK*

Aqueles que trabalham com SQL regularmente deveriam se tornar familiar com ambos as classes antigas (SQL92) e as novas (SQL3), pelo fato que ambas nomenclaturas ainda são usadas para se referir aos atributos e instruções SQL.

Dialetos do SQL

A natureza constantemente envolvente do padrão SQL deu um aumento para vários dialetos SQL entre os vários fornecedores e plataformas. Estes dialetos comumente evoluem, pois um determinado banco de dados de uma comunidade de usuário do fornecedor requer habilidades no banco de dados antes do comitê ANSI criar um padrão aplicável. Ocasionalmente, porém, as comunidades acadêmicas e de pesquisa introduzem um atributo novo em resposta a pressões de tecnologias concorrentes. Por exemplo, muitos fornecedores de banco de dados estão aumentando suas recentes ofertas programáticas tanto com Java (como no caso com DB2, Oracle e Sybase) quanto VBScript (como no caso com Microsoft). No futuro, programadores e desenvolvedores usarão estas linguagens de programação em conjunto com SQL para construir programas SQL.

Muitos destes dialetos incluem habilidades de processamento condicional (assim como aquelas controladas por instruções *IF...THEN*), funções de controle de fluxo (assim como laços *WHILE*), variáveis e habilidades de manipulação de erros. Porque ANSI não desenvolveu ainda um padrão para estes atributos importantes no tempo que usuários começam a exigir que eles, desenvolvedores e fornecedores de SGBDR criaram seus próprios comandos e sintaxe. De fato, alguns dos fornecedores mais antigos dos anos 80 tem variâncias nos comandos mais elementares, como *SELECT*, pois suas implementações antecedem os padrões. ANSI está agora refinando padrões que endereçam estas inconsistências.

Alguns destes dialetos introduziram comandos procedurais para suportar a funcionalidade de uma linguagem de programação mais completa. Por exemplo, estas implementações procedural contêm comandos para lidar com erros, linguagem de controle de fluxo, comandos condicionais, comandos para manipular variáveis, suporte para arrays e muitas outras extensões. Embora estas sejam implementações procedurais tecnicamente divergentes, são chamadas dialetos aqui. O pacote SQL/PSM (Persitent Stored Module) fornece muitos atributos associados com procedimentos armazenados de programação e incorpora muitas das extensões oferecidas por estes dialetos.

Alguns dialetos populares de SQL incluem:

PL/SQL
: Encontrado no Oracle. PL/SQL significa Procedural Language/SQL e contém muitas similaridades com a linguagem Ada.

Transact-SQL
: Usados por ambos Microsoft SQL Server e Sybase Adaptive Server como Microsoft e Sybase mudaram de uma plataforma comum que eles compartilhavam no inicio dos anos 90, suas implementações de Transact-SQL também divergiram.

PL/pgSQL
: Dialeto e extensões SQL implementadas no PostgreSQL. As iniciais significam Procedural Language/PostgreSQL.

Usuários que planejam trabalhar extensivamente com um único sistema de banco de dados deveriam aprender as complicações de seus dialetos ou plataformas SQL preferidas.