Elmasri • Navathe

Sistemas de banco de dados

6ª Edição

PEARSON

Companion Website

5.2 Especificando restrições como asserções e ações como triggers

Nesta seção, apresentamos dois recursos adicionais da SQL: o comando **CREATE ASSERTION** e o comando **CREATE TRIGGER**. A Seção 5.2.1 discute o CREATE ASSERTION, que pode ser usado para especificar tipos adicionais de restrições que estão fora do escopo das *restrições embutidas do modelo relacional* (chaves primária e única, integridade de entidade e integridade referencial), que apresentamos na Seção 3.2. Essas restrições embutidas podem ser especificadas dentro do comando **CREATE TABLE** da SQL (ver seções 4.1 e 4.2).

Depois, na Seção 5.2.2, apresentamos CREATE TRIGGER, que pode ser usado para especificar ações automáticas que o sistema de banco de dados realizará quando certos eventos e condições ocorrerem. Esse tipo de funcionalidade costuma ser conhecido como **bancos de dados ativos**. Só apresentamos os fundamentos básicos dos **triggers** neste capítulo, e uma discussão mais completa sobre os bancos de dados ativos pode ser encontrada na Seção 26.1.

5.2.1 Especificando restrições gerais como asserções em SQL

Em SQL, os usuários podem especificar restrições gerais — aquelas que não se encaixam em nenhuma das categorias descritas nas seções 4.1 e 4.2 — por meio de **asserções declarativas**, usando o comando CREATE ASSERTION da DDL. Cada asserção recebe um nome de restrição e é especificada por uma condição semelhante à cláusula WHERE de uma consulta SQL. Por exemplo, para especificar a restrição de que *o salário de um funcionário não pode ser maior que o salário do gerente do departamento para o qual o funcionário trabalha* em SQL, podemos escrever a seguinte asserção:

```
CREATE ASSERTION RESTRICAO_SALARIAL
CHECK ( NOT EXISTS
        ( SELECT    *
        FROM    FUNCIONARIO F,
                FUNCIONARIO G,
                DEPARTAMENTO D
        WHERE   F.Salario>G.Salario
                AND F.Dnr=D.Dnumero
                AND D. Cpf_gerente =G.Cpf ) );
```

O nome de restrição RESTRICAO_SALARIAL é seguido pela palavra-chave CHECK, que é seguida por uma **condição** entre parênteses que precisa ser verdadeira em cada estado do banco de dados para que a asserção seja satisfeita. O nome da restrição pode ser usado mais tarde para se referir à restrição ou para modificá-la ou excluí-la. O SGBD é responsável por garantir que a condição não seja violada. Qualquer condição de cláusula WHERE pode ser usada, mas muitas restrições podem ser especificadas usando o estilo EXISTS e NOT EXISTS das condições em SQL. Sempre que alguma tupla no banco de dados fizer que a condição de um comando ASSERTION seja avaliada como FALSE, a restrição é **violada**. A restrição é **satisfeita** por um estado do banco de dados se *nenhuma combinação de tuplas* nesse estado do banco de dados violar a restrição.

A técnica de uso comum para escrever essas asserções é especificar uma consulta que seleciona quaisquer tuplas *que violam a condição desejada*. Ao incluir essa consulta em uma cláusula NOT EXISTS, a asserção especificará que o resultado dessa consulta precisa ser vazio para que a condição seja sempre TRUE. Assim, uma asserção é violada se o resultado da consulta não for vazio. No exemplo anterior, a consulta seleciona todos os funcionários cujos salários são maiores que o salário do gerente de seu departamento. Se o resultado da consulta não for vazio, a asserção é violada.

Observe que a cláusula CHECK e a condição de restrição também podem ser utilizadas para especificar restrições sobre atributos e domínios *individuais* (ver Seção 4.2.1) e sobre tuplas *individuais* (ver Seção 4.2.4). A principal diferença entre CREATE ASSERTION e as restrições de domínio e de tupla individuais é que as cláusulas CHECK sobre atributos, domínios e tuplas individuais são verificadas na SQL *somente quando as tuplas são inseridas ou atualizadas*. Logo, a verificação de restrição pode ser implementada de maneira mais eficiente pelo SGBD nesses casos. O projetista do esquema deve usar CHECK sobre atributos, domínios e tuplas apenas quando estiver certo de que a restrição *só pode ser violada pela inserção ou atualização de tuplas*. Além disso, o projetista do esquema deve usar CREATE ASSERTION somente em casos em que não é possível usar CHECK sobre atributos, domínios ou tuplas, de modo que verificações simples são implementadas de modo mais eficiente pelo SGBD.

5.2.2 Introdução às triggers em SQL

Outro comando importante em SQL é o CREATE TRIGGER. Em muitos casos, é conveniente especificar um tipo de ação a ser tomada quando certos eventos ocorrem e quando certas condições são satisfeitas. Por exemplo, pode ser útil especificar uma condição que, se violada, faz que algum usuário seja informado dela. Um gerente pode querer ser informado se as despesas de viagem de um funcionário excederem certo limite, recebendo uma mensagem sempre que isso acontecer. A ação que o SGBD deve tomar nesse caso é enviar uma mensagem apropriada a esse usuário. A condição, portanto, é usada para **monitorar**

o banco de dados. Outras ações podem ser especificadas, como executar um procedimento armazenado (*stored procedure*) específico ou disparar outras atualizações. A instrução CREATE TRIGGER é utilizada para implementar essas ações em SQL. Discutiremos sobre triggers (gatilhos) com detalhes na Seção 26.1, quando descreveremos os *bancos de dados ativos*. Aqui, vamos apenas dar um exemplo simples de como os triggers podem ser usadas.

Suponha que queiramos verificar se o salário de um funcionário é maior que o salário de seu supervisor direto no banco de dados EMPRESA (ver figuras 3.5 e 3.6). Vários eventos podem disparar essa regra: inserir um novo registro de funcionário, alterar o salário de um funcionário ou alterar o supervisor de um funcionário. Suponha que a ação a ser tomada seria chamar o procedimento armazenado,[5] que notificará o supervisor. O trigger poderia então ser escrita como em R5, a seguir. Aqui, estamos usando a sintaxe do sistema de banco de dados Oracle.

```
R5: CREATE TRIGGER VIOLACAO_SALARIAL
    BEFORE INSERT OR UPDATE OF SALARIO,
    CPF_SUPERVISOR ON FUNCIONARIO
FOR EACH ROW
    WHEN ( NEW.SALARIO >
        ( SELECT SALARIO FROM FUNCIONARIO
        WHERE CPF = NEW.CPF_SUPERVISOR ) )
        INFORMAR_SUPERVISOR
        (NEW.Cpf_supervisor,
        NEW.Cpf );
```

O trigger recebe o nome VIOLACAO_SALARIAL, que pode ser usada para remover ou desativar o trigger mais tarde. Um trigger típico tem três componentes:

1. O(s) **evento(s)**: estes em geral são operações de atualização no banco de dados, aplicadas explicitamente a ele. Neste exemplo, os eventos são: inserir um novo registro de funcionário, alterar o salário de um funcionário ou alterar o supervisor de um funcionário. A pessoa que escreve o trigger precisa garantir que todos os eventos possíveis sejam considerados. Em alguns casos, pode ser preciso escrever mais de um trigger para cobrir todos os casos possíveis. Esses eventos são especificados após a palavra-chave **BEFORE** em nosso exemplo, o que significa que o trigger deve ser executada antes que a operação de disparo seja executada. Uma alternativa é usar a palavra-chave **AFTER**, que especifica que o trigger deve ser executada após a operação especificada no evento ser concluída.
2. A **condição** que determina se a ação da regra deve ser executada: depois que o evento de disparo tiver ocorrido, uma condição *opcional* pode ser avaliada. Se *nenhuma condição* for especificada, a ação será executada uma vez que o evento ocorra. Se uma condição for especificada, ela primeiro é avaliada e, somente *se for avaliada como verdadeira*, a ação da regra será executada. A condição é especificada na cláusula WHEN do trigger.
3. A **ação** a ser tomada: a ação normalmente é uma sequência de instruções em SQL, mas também poderia ser uma transação de banco de dados ou um programa externo que será executado automaticamente. Neste exemplo, a ação é executar o procedimento armazenado INFORMAR_SUPERVISOR.

Os triggers podem ser usados em várias aplicações, como na manutenção da coerência do banco de dados, no monitoramento de atualizações do banco de dados e na atualização de dados derivados automaticamente. Uma discussão mais completa pode ser vista na Seção 26.1.

5.3 Visões (views) — Tabelas virtuais em SQL

Nesta seção, apresentamos o conceito de uma view (visão) em SQL. Mostraremos como as views são especificadas, depois discutiremos o problema de atualizá-las e como elas podem ser implementadas pelo SGBD.

5.3.1 Conceito de uma view em SQL

Uma view em terminologia SQL é uma única tabela que é derivada de outras tabelas.[6] Essas outras tabelas podem ser *tabelas da base* ou views previamente definidas. Uma view não necessariamente existe em forma física; ela é considerada uma **tabela virtual**, ao contrário das **tabelas da base**, cujas tuplas sempre estão armazenadas fisicamente no banco de dados. Isso limita as possíveis operações de atualização que podem ser aplicadas às views, mas não oferece quaisquer limitações sobre a consulta de uma view.

[5] Supondo que um procedimento externo tenha sido declarado. Discutiremos os procedimentos armazenados no Capítulo 13.

[6] Conforme usado em SQL, o termo *view* é mais limitado do que o termo *view do usuário* discutido nos capítulos 1 e 2, pois esta última possivelmente incluiria muitas relações.