
ORACLE
Academy

Programação de Banco de Dados com SQL

12-3
Valores DEFAULT, MERGE e Instruções INSERT em Várias Tabelas

ORACLE Academy

Objetivos

Nesta lição, você aprenderá a:

- Compreender quando deve especificar um valor DEFAULT
- Construir e executar uma instrução MERGE
- Construir e executar instruções DML usando subconsultas
- Construir e executar instruções INSERT em várias tabelas

Finalidade

- Até o momento, você atualizou dados usando uma única instrução INSERT.
- Isso é relativamente fácil quando se está adicionando um registro por vez, mas e se a sua empresa for muito grande e utilizar data warehouse para armazenar registros de vendas e dados pessoais, de clientes, de folhas de pagamentos e de contabilidade?
- Nesse caso, os dados provavelmente estão vindo de vários lugares e sendo gerenciados por várias pessoas.
- O gerenciamento de um registro de dados por vez poderia ser algo muito confuso e demorado.

Finalidade

- Como você determina o que foi inserido ou alterado recentemente?
- Nesta lição, você aprenderá um método mais eficiente de atualizar e inserir dados usando uma sequência de comandos INSERT e UPDATE condicionais em uma única instrução atômica.
- Você também aprenderá a recuperar dados de uma única subconsulta e inserir as linhas retornadas em mais de uma tabela de destino. descobrirá maneiras eficazes de fazer seu trabalho.
- À medida que aumenta seu conhecimento de SQL, você

DEFAULT

- Cada coluna em uma tabela pode ter um valor padrão especificado.
- No caso de uma nova linha ser inserida e não ser atribuído um valor para a coluna, o valor padrão será atribuído no lugar do valor nulo.
- O uso de valores padrão permite controlar onde e quando eles serão aplicados.

DEFAULT

- O valor padrão pode ser um valor literal, uma expressão ou uma função SQL, como SYSDATE e USER, mas não pode ser o nome de outra coluna.
- O valor padrão deve ser equivalente ao tipo de dados da coluna.
- DEFAULT pode ser especificado para uma coluna quando a tabela é criada ou alterada.

Exemplo de DEFAULT

- O exemplo abaixo mostra um valor padrão sendo especificado para a coluna hire_date no momento em que a tabela é criada:

```
CREATE TABLE my_employees (
       hire_date  DATE DEFAULT SYSDATE,
       first_name   VARCHAR2(15),
       last_name  VARCHAR2(15));
```

- Quando linhas forem adicionadas a essa tabela, SYSDATE será atribuído a qualquer linha que não especifique explicitamente um valor de hire_date.

DEFAULT Explícito com INSERT

- Valores padrão explícitos podem ser usados em instruções INSERT e UPDATE.
- O exemplo de INSERT utilizando a tabela my_employees mostra o uso explícito de DEFAULT:

```
INSERT INTO my_employees
 (hire_date, first_name, last_name)
VALUES
 (DEFAULT, 'Angelina','Wright');
```

- Uso implícito de DEFAULT

```
INSERT INTO my_employees
 (first_name, last_name)
VALUES
 ('Angelina','Wright');
```

DEFAULT Explícito com UPDATE

- Valores padrão explícitos podem ser usados em instruções INSERT e UPDATE.
- O exemplo de UPDATE utilizando a tabela my_employees mostra o uso explícito de DEFAULT.

```
UPDATE my_employees
SET hire_date = DEFAULT
WHERE last_name = 'Wright';
```

- Se um valor padrão for especificado para a coluna hire_date, a coluna recebe o valor padrão.
- No entanto, se um valor padrão não for especificado quando a coluna for criada, um valor nulo será atribuído.

MERGE

- O uso da instrução MERGE realiza duas tarefas ao mesmo tempo. MERGE faz uma inserção e atualização simultaneamente. Se um valor estiver faltando, um novo será inserido.
- Se um valor existir, mas precisar ser alterado, MERGE vai atualizá-lo.
- Para fazer esses tipos de alterações nas tabelas de bancos de dados, você precisa ter privilégios para usar INSERT e UPDATE na tabela de destino e para usar SELECT na tabela de origem.
- É possível usar aliases com a instrução MERGE.

Sintaxe de MERGE

- Uma linha por vez é lida na tabela de origem e comparada às linhas na tabela de destino usando a condição de equivalência.
- Se houver uma linha correspondente na tabela de destino, a linha de origem é usada para atualizar uma ou mais colunas na linha de destino correspondente.
- Se não houver uma linha correspondente, os valores da linha de origem são usados para inserir uma nova linha na tabela de destino.

```
MERGE INTO tabela-destino USING tabela-origem
  ON condição-equivalência
WHEN MATCHED THEN UPDATE
  SET ……
WHEN NOT MATCHED THEN INSERT
  VALUES (……);
```

Exemplo de MERGE

- Este exemplo usa a tabela EMPLOYEES (alias e) como a origem dos dados para inserir e atualizar linhas em uma cópia da tabela chamada COPY_EMP (alias c).

```
MERGE INTO copy_emp c  USING employees e
  ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN UPDATE
   SET
    c.last_name = e.last_name,
   c.department_id = e.department_id
WHEN NOT MATCHED THEN INSERT
   VALUES (e.employee_id, e.last_name, e.department_id);
```

Exemplo de MERGE

- As linhas 100 e 103 de EMPLOYEES tinham linhas correspondentes em COPY_EMP. Portanto, as linhas correspondentes de COPY_EMP foram atualizadas.
- A linha 142 de EMPLOYEES não tinha uma linha correspondente. Portanto, foi inserida em COPY_EMP.

EMPLOYEES (tabela de origem)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

Exemplo de MERGE

EMPLOYEES (tabela de origem)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

COPY_EMP antes de MERGE ser executado

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	Smith	40
103	Chang	30

COPY_EMP depois de MERGE ser executado

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

Instruções INSERT em Várias Tabelas

- As instruções INSERT em várias tabelas são usadas quando os mesmos dados de origem devem ser inseridos em mais de uma tabela de destino.
- Essa funcionalidade é útil quando você está trabalhando em um ambiente de data warehouse em que é comum mover dados regularmente dos sistemas operacionais para o data warehouse para geração de relatórios e análise.
- A criação e o gerenciamento de data warehouses é uma maneira de administrar o às vezes grande número de linhas inseridas nos sistemas operacionais durante um dia de trabalho normal.

Instruções INSERT em Várias Tabelas

- Imagine, por exemplo, quantas linhas de dados a sua operadora de telefone deve criar diariamente para todas
  as ligações ou mensagens de texto feitas em todos os dispositivos aos quais você tem acesso.
- Acrescente a isso a navegação na Internet e o download de toques, papéis de parede, jogos e outros aplicativos para dispositivos móveis.
- Multiplique esse número pelo total de clientes e você talvez tenha uma ideia da quantidade de dados que as empresas de telecomunicação precisam gerenciar.

Instruções INSERT em Várias Tabelas

- É possível que essas linhas tenham que ser inseridas em mais de uma tabela no data warehouse. Logo, se conseguirmos selecioná-las apenas uma vez e replicá-las, o desempenho será melhor.
- As instruções INSERT em várias tabelas podem ser condicionais ou incondicionais. Em uma instrução INSERT em várias tabelas incondicional, o Oracle insere todas as linhas retornadas pela subconsulta em todas as cláusulas de inserção de tabela encontradas na instrução.
- Em uma instrução INSERT em várias tabelas condicional, você pode especificar ALL ou FIRST.

Instruções INSERT em Várias Tabelas

- Especificando ALL:
  - Se você especificar ALL, o valor padrão, o banco de dados avalia cada cláusula WHEN, independentemente dos resultados da avaliação de qualquer outra cláusula WHEN.
  - Para cada cláusula WHEN cuja condição seja avaliada como verdadeira, o banco de dados executa a lista da cláusula INTO correspondente.
- Especificando FIRST:
  - Se você especificar FIRST, o banco de dados avalia cada cláusula WHEN na ordem em que aparece na instrução.
  - Para a primeira cláusula WHEN que for avaliada como verdadeira, o banco de dados executa a cláusula INTO correspondente e ignora as cláusulas WHEN subsequentes para a linha fornecida.

Instruções INSERT em Várias Tabelas

- Especificando a cláusula ELSE:
- Para uma determinada linha, se nenhuma cláusula WHEN for avaliada como verdadeira, o banco de dados executa a lista da cláusula INTO associada à cláusula ELSE.
- Se você não especificou uma cláusula ELSE, o banco de dados não executará uma ação para a linha.

Sintaxe da Instrução INSERT em Várias Tabelas

- A sintaxe da instrução INSERT em várias tabelas é a seguinte:

```
INSERT ALL cláusula INTO cláusula VALUES SUBCONSULTA
```

- Eis um exemplo da instrução INSERT em várias tabelas:

```
INSERT ALL
 INTO my_employees
  VALUES (hire_date, first_name, last_name)
 INTO copy_my_employees
  VALUES (hire_date, first_name, last_name)
SELECT hire_date, first_name, last_name
FROM employees;
```

Instruções INSERT em Várias Tabelas Condicionais

```
INSERT ALL
 WHEN call_ format IN ('tlk','txt','pic') THEN
 INTO all_calls
  VALUES (caller_id, call_timestamp, call_duration, call_format)
 WHEN call_ format IN ('tlk','txt') THEN
 INTO police_record_calls
  VALUES (caller_id, call_timestamp, recipient_caller)
 WHEN call_duration < 50 AND call_type = 'tlk' THEN
 INTO short_calls
  VALUES (caller_id, call_timestamp, call_duration)
 WHEN call_duration > = 50 AND call_type = 'tlk' THEN
 INTO long_calls
  VALUES (caller_id, call_timestamp, call_duration)
SELECT  caller_id, call_timestamp, call_duration, call_format,
    recipient_caller
FROM calls
WHERE TRUNC(call_timestamp ) = TRUNC(SYSDATE);
```

Terminologia

Estes são os principais termos usados nesta lição:

- DEFAULT
- MERGE
- Instruções INSERT em Várias Tabelas
- ALL, FIRST e ELSE

Resumo

Nesta lição, você deverá ter aprendido a:

- Compreender quando deve especificar um valor DEFAULT
- Construir e executar uma instrução MERGE
- Construir e executar instruções DML usando subconsultas
- Construir e executar instruções INSERT em várias tabelas


ORACLE
Academy