FIAP GRADUAÇÃO

FIAP

Tecnologia em Análise e Desenvolvimento de Sistemas

Application Development For Databases

PROF. MILTON

Versão 1 – <agosto de 25>

Criando Triggers

Objetivos

Ao concluir esta lição, você será capaz de:

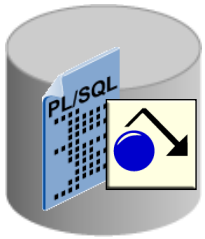
- Descrever triggers de banco de dados e seu uso
- Descrever os diferentes tipos de triggers
- Criar triggers de banco de dados
- Descrever regras de acionamento de triggers de banco de dados
- Remover triggers de banco de dados
- Exibir informações sobre triggers

Objetivo da Lição

Nesta lição, você aprenderá a criar e usar triggers de banco de dados.

O Que São Triggers?

- Um trigger é um bloco PL/SQL armazenado no banco de dados e acionado (executado) em resposta a um evento especificado.
- O Oracle database executa automaticamente um trigger quando condições específicas ocorrem.

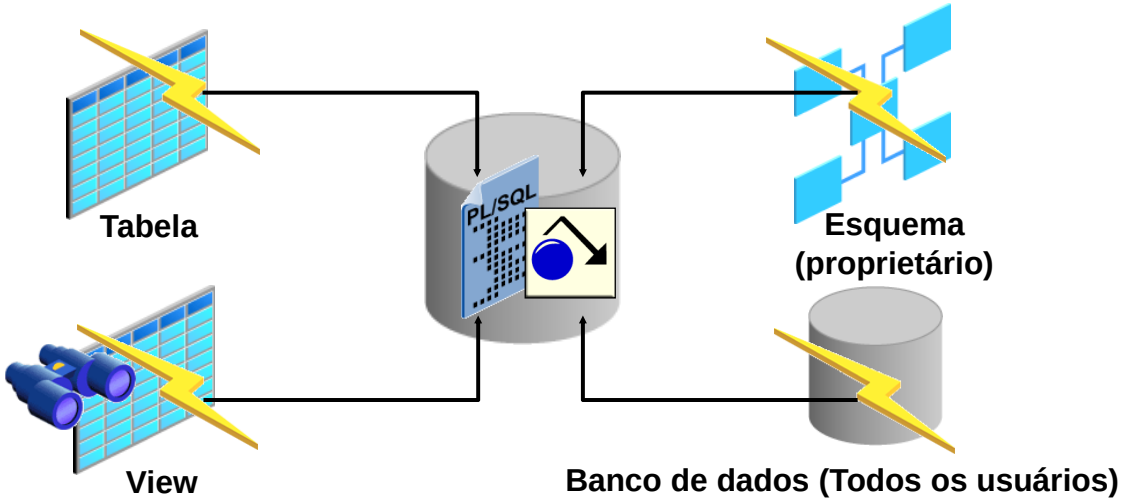


Trabalhando com Triggers: Visão Geral

Os triggers são semelhantes a procedimentos armazenados. Um trigger armazenado no banco de dados contém PL/SQL na forma de um bloco anônimo, uma instrução de chamada ou um bloco de trigger composto. Entretanto, os procedures e triggers diferem na forma como são chamados. Um procedure é executado explicitamente por um usuário, uma aplicação ou trigger. Os triggers são acionados implicitamente pelo Oracle database quando um evento de trigger ocorre, não importando qual usuário está conectado ou qual aplicação está sendo usada.

Definindo Triggers

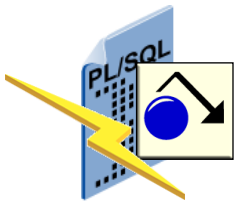
Um trigger pode ser definido na tabela, view, esquema (proprietário do esquema) ou banco de dados (todos os usuários).



Tipos de Eventos de Trigger

Você pode gravar triggers que são acionados sempre que uma das seguintes operações ocorra no banco de dados:

- Uma instrução DML (database manipulation) (`DELETE`, `INSERT`, ou `UPDATE`).
- Uma instrução DDL (database definition) (`CREATE`, `ALTER`, ou `DROP`).
- Uma operação em banco de dados como `SERVERERROR`, `LOGON`, `LOGOFF`, `STARTUP`, ou `SHUTDOWN`.



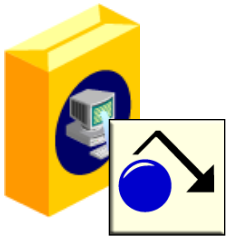
Evento ou Instrução de Trigger

Um evento ou instrução de trigger é a instrução SQL, evento de banco de dados ou evento de usuário que faz com que um trigger seja acionado. Um evento de trigger pode ser um ou mais dos seguintes:

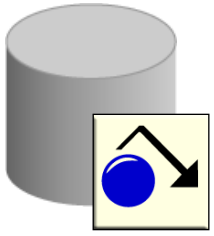
- Uma instrução `INSERT`, `UPDATE`, ou `DELETE` em uma tabela específica (ou view, em alguns casos)
- Uma instrução `CREATE`, `ALTER`, ou `DROP` em qualquer objeto de esquema
- Um shutdown de instância ou inicialização de banco de dados
- Uma mensagem de erro específica ou qualquer mensagem de erro
- Um logon ou logoff de usuário

Triggers de Aplicação e Banco de Dados

- Trigger de banco de dados (abordado neste curso):
  - É acionado sempre que um evento DML, DLL ou de sistema ocorre em um esquema ou banco de dados
- Trigger de aplicação:
  - É disparado sempre que ocorre um evento em determinada aplicação



Trigger de aplicação



Trigger de Banco de Dados

Tipos de Triggers

Os triggers de aplicação são executados implicitamente sempre que ocorre um evento DML (Data Manipulation Language) em uma aplicação. Por exemplo, as aplicações desenvolvidas com o Oracle Forms Developer fazem amplo uso de triggers.

Os triggers de banco de dados são executados implicitamente quando ocorre algum dos seguintes eventos:

- Operações DML em uma tabela
- Operações DML em uma view, com um trigger `INSTEAD OF`
- Instruções DDL, como `CREATE` e `ALTER`

Isso se aplicará independentemente do usuário que estiver conectado ou da aplicação que estiver sendo usada. Os triggers de banco de dados também são executados de forma implícita quando ocorrem algumas ações do usuário ou do sistema de banco de dados (por exemplo, quando um usuário efetua logon ou o DBA faz shutdown no banco de dados).

Os triggers de banco de dados podem ser triggers do sistema em um banco de dados ou em um esquema (abordado na próxima lição). No caso de bancos de dados, os triggers são disparados em cada evento para todos os usuários. No caso de um esquema, eles são disparados em cada evento desse usuário específico. O Oracle Forms pode definir, armazenar e executar triggers de classificações diferentes. Entretanto, não confunda os triggers do Oracle Forms com os triggers abordados nesta lição.

Cenários de Aplicações de Negócios para a Implementação de Triggers

Você pode usar triggers para:

- Segurança
- Auditoria
- Integridade dos dados
- Integridade referencial
- Replicação de tabelas
- Cálculo automático de dados derivados
- Log de eventos

Cenários de Aplicações de Negócios para a Implementação de Triggers

Desenvolva triggers de banco de dados para aprimorar os recursos que não podem ser implementados pelo servidor Oracle ou como alternativas aos fornecidos por ele.

- **Segurança:** O servidor Oracle permite o acesso de usuários ou atribuições a tabelas. Os triggers permitem o acesso a tabelas de acordo com os valores de dados.
- **Auditoria:** O servidor Oracle controla as operações de dados em tabelas. Os triggers controlam os valores dessas operações em tabelas.
- **Integridade dos dados:** O servidor Oracle impõe constraints de integridade, e os triggers implementam regras complexas de integridade.
- **Integridade referencial:** O servidor Oracle impõe regras-padrão de integridade referencial, e os triggers implementam funcionalidades não-padrão.
- **Replicação de tabelas:** O servidor Oracle copia tabelas em snapshots de forma assíncrona, enquanto os triggers as copiam em réplicas de forma síncrona.
- **Dados derivados:** O servidor Oracle calcula os valores dos dados derivados manualmente, enquanto os triggers calculam esses valores de forma automática.
- **Log de eventos:** O servidor Oracle registra os eventos em log de forma explícita, enquanto os triggers os registram de forma transparente.

Tipos de Trigger Disponíveis

- Triggers DML simples
  - BEFORE
  - AFTER
  - INSTEAD OF
- Triggers compostos
- Triggers não DML
  - Triggers de evento DDL
  - Triggers de evento de banco de dados

Observação

Nesta lição, abordaremos os triggers BEFORE, AFTER e INSTEAD OF. Os outros tipos de trigger são abordados na lição intitulada "Criando Triggers Compostos, de DDL e de Banco de Dados de Eventos."

Tipos de Eventos de Trigger e Trigger Body

- Um tipo de evento de trigger determina qual instrução DML faz com que o trigger seja executado. Os eventos possíveis são:
  - `INSERT`
  - `UPDATE [OF column]`
  - `DELETE`
- Um trigger body determina qual ação é executada e é um bloco PL/SQL ou uma `CALL` a um procedure.

Tipos de Eventos de Trigger

O evento ou instrução de trigger pode ser uma instrução `INSERT`, `UPDATE` ou `DELETE` em uma tabela.

- Quando o evento de trigger é uma instrução `UPDATE`, você pode incluir uma lista de colunas para identificar as colunas que devem ser alteradas para que o trigger seja disparado. Não é possível especificar uma lista de colunas para uma instrução `INSERT` ou `DELETE` porque essas instruções sempre afetam linhas inteiras.

```
. . . UPDATE OF salary . . .
```

- O evento de trigger pode conter uma, duas ou todas as três operações DML.

```
. . . INSERT or UPDATE or DELETE
. . . INSERT or UPDATE OF job_id . . .
```

O trigger body define a ação, ou seja, o que é necessário fazer quando o evento de trigger é executado. O bloco PL/SQL pode conter instruções SQL e PL/SQL, bem como definir estruturas PL/SQL, como variáveis, cursores, exceções etc. Você também pode chamar um procedure PL/SQL ou Java.

Observação: O tamanho máximo de um trigger não pode exceder 32 KB.

Criando Triggers DML Usando a Instrução CREATE TRIGGER

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing -- when to fire the trigger
event1 [OR event2 OR event3]
ON object_name
[REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW -- default is statement level trigger
WHEN (condition)]]
DECLARE]
BEGIN
... trigger_body -- executable statements
[EXCEPTION . . .]
END [trigger_name];
```

```
timing =  BEFORE | AFTER | INSTEAD OF
```

```
event = INSERT | DELETE | UPDATE | UPDATE OF column_list
```

Criando Triggers DML

Estes são os componentes da sintaxe do trigger:

- *trigger_name* identifica o trigger de forma exclusiva.
- *timing* indica quando o trigger é disparado em relação ao evento de trigger. Os valores são BEFORE, AFTER e INSTEAD OF.
- event identifica a operação DML responsável pelo acionamento do trigger. Os valores são INSERT, UPDATE [OF column] e DELETE.
- *object_name* indica a tabela ou a view associada ao trigger.
- Para triggers de linha, é possível especificar:
  - Uma cláusula REFERENCING para escolher os nomes de correlação usados ao fazer referência aos valores antigos e novos da linha atual (os valores default são OLD e NEW)
  - FOR EACH ROW para designar que o trigger é de linha
  - Uma cláusula WHEN para aplicar um predicado condicional, entre parênteses, que é avaliado em cada linha para determinar se o trigger body deve ou não ser executado
- O *trigger_body* é a ação executada pelo trigger, implementada como um dos seguintes itens:
  - Um bloco anônimo com uma instrução DECLARE ou BEGIN e uma instrução END
  - Uma cláusula CALL para chamar um procedure armazenado stand-alone ou encapsulado, como:

    ```
    CALL my_procedure;
    ```

Especificando o Acionamento do Trigger (Tempo de Execução)

Você pode especificar o momento de execução do trigger, disparando a ação do trigger antes ou depois da instrução responsável pelo seu acionamento:

- `BEFORE`: Executa o trigger body antes do evento DML de acionamento do trigger em uma tabela.
- `AFTER`: Executa o trigger body depois do evento DML de trigger em uma tabela.
- `INSTEAD OF`: Executa o trigger body em vez da instrução responsável pelo acionamento do trigger. Essa opção é usada em views que não podem ser modificadas de outra maneira.

Momento de Acionamento do Trigger

Em geral, os triggers do tipo **`BEFORE`**são usados nas seguintes situações:

- Para determinar se a instrução responsável pelo acionamento do trigger deve ser concluída. (Isso elimina o processamento desnecessário e permite o rollback nos casos em que uma exceção é gerada na ação do trigger.)
- Para derivar valores de coluna antes de concluir uma instrução `INSERT` ou `UPDATE`
- Para inicializar variáveis ou flags globais e para validar regras de negócios complexas

Em geral, os triggers do tipo **`AFTER`**são usados nas seguintes situações:

- Para concluir a instrução que aciona o trigger antes de executar a ação do trigger
- Para executar diversas ações na mesma instrução que aciona o trigger se um trigger `BEFORE` já existir

Os triggers do tipo **`INSTEAD OF`** proporcionam uma forma transparente de modificar views que não podem ser modificadas diretamente por meio de instruções SQL DML, pois as views nem sempre são modificáveis. Você pode criar instruções DML adequadas dentro do corpo de um trigger `INSTEAD OF` para executar ações diretamente nas tabelas subjacentes das views.

Se for prático, substitua o conjunto de triggers individuais com diferentes momentos de sincronização por um trigger único composto que codifica explicitamente as ações na ordem pretendida. Se dois ou mais triggers forem definidos com o mesmo ponto de sincronização, e a ordem na qual forem acionados for importante, então você pode controlar a ordem de acionamento usando as cláusulas `FOLLOWS` e `PRECEDES`.

Triggers de Nível de Instrução Versus Triggers de Nível de Linha

Triggers de Nível de Instrução	Triggers de Nível de Linha
É o default ao se criar um trigger	Use a cláusula `FOR EACH ROW` ao criar um trigger.
É executado uma vez para o evento de trigger	É executado uma vez para cada linha afetada pelo evento de trigger
É disparado uma vez mesmo que nenhuma linha seja afetada	Não será executado se nenhuma linha for afetada pelo evento de trigger

Tipos de Triggers DML

Você pode especificar que o trigger seja executado uma vez para cada linha afetada pela instrução responsável pelo acionamento do trigger (como uma operação `UPDATE` de várias linhas) ou uma vez para essa instrução, independentemente do número de linhas afetadas.

Trigger de Instrução

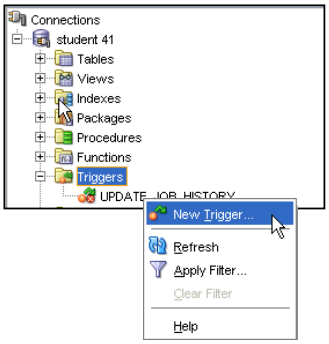
O trigger de instrução é disparado uma vez para o evento de trigger, mesmo que nenhuma linha seja afetada. Esse tipo de trigger será útil se a ação do trigger não depender dos dados das linhas afetadas ou dos dados fornecidos pelo próprio evento de trigger (por exemplo, um trigger que executa uma verificação de segurança complexa no usuário atual).

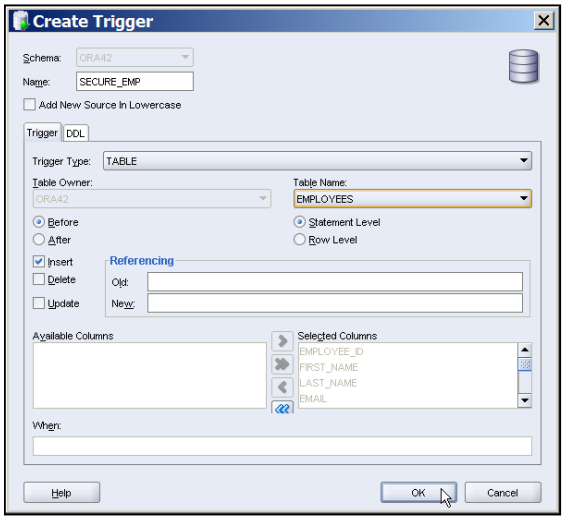
Trigger de Linha

O trigger de linha é disparado toda vez que a tabela é afetada pelo evento de trigger. Se o evento não afetar qualquer linha, o trigger de linha não será executado. Esse tipo de trigger será útil se a ação do trigger depender dos dados das linhas afetadas ou dos dados fornecidos pelo próprio evento.

Observação: Os triggers de linha usam nomes de correlação para acessar os valores de coluna antigos e novos da linha que está sendo processada pelo trigger.

Criando Triggers DML Usando o SQL Developer



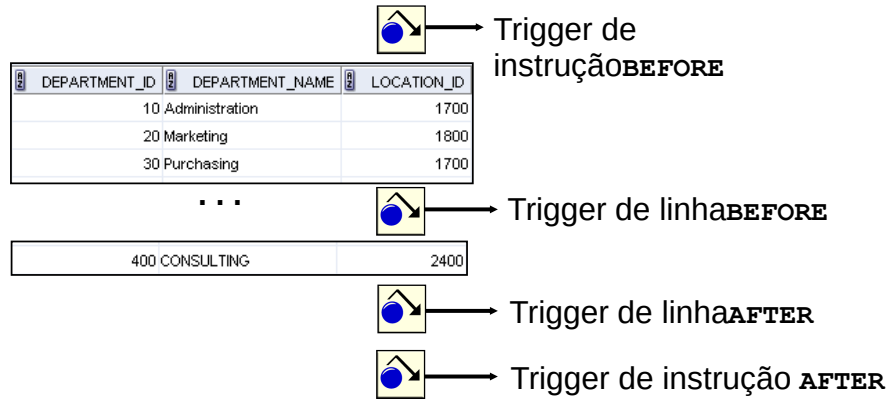



```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON employees BEGIN
 IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
      (TO_CHAR(SYSDATE,'HH24:MI') NOT BETWEEN '08:00' AND '18:00') THEN
  RAISE_APPLICATION_ERROR(-20500, 'You may insert' ||
   ' into EMPLOYEES table only during ' ||' business hours.');
  END IF;
END;
```

Sequência de Acionamento de Trigger: Manipulação de uma Única Linha

Use a seguinte sequência para disparar um trigger em uma tabela quando uma única linha for manipulada:

```
INSERT INTO departments
   (department_id,department_name, location_id)
VALUES (400, 'CONSULTING', 2400);
```



Sequência de Acionamento de Triggers: Manipulação de uma Única Linha

Crie um trigger de instrução ou um trigger de linha, de acordo com as necessidades, ou seja, se é necessário disparar o trigger uma vez para cada linha afetada pela instrução responsável por seu acionamento ou somente uma vez para essa instrução, independentemente do número de linhas afetadas.

Quando a instrução DML que aciona o trigger afeta uma única linha, os triggers de instrução e de linha são disparados exatamente uma vez.

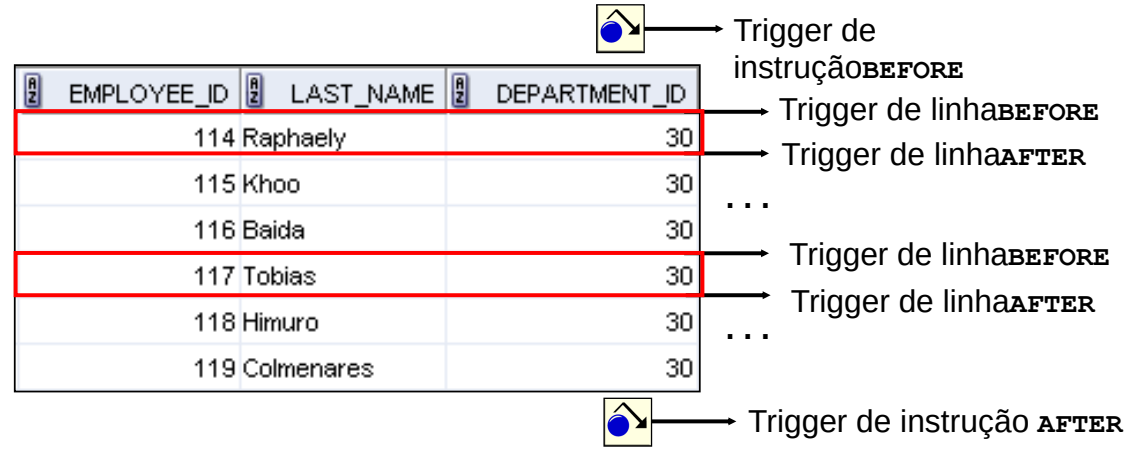
Exemplo

A instrução SQL do slide não faz distinção entre triggers de instrução e triggers de linha, pois exatamente uma linha é inserida na tabela por meio da sintaxe da instrução `INSERT` mostrada no slide..

Sequência de Acionamento de Triggers: Manipulação de Várias Linhas

Use a seguinte sequência para disparar um trigger em uma tabela quando várias linhas forem manipuladas:

```
UPDATE employees
  SET salary = salary * 1.1
  WHERE department_id =30;
```



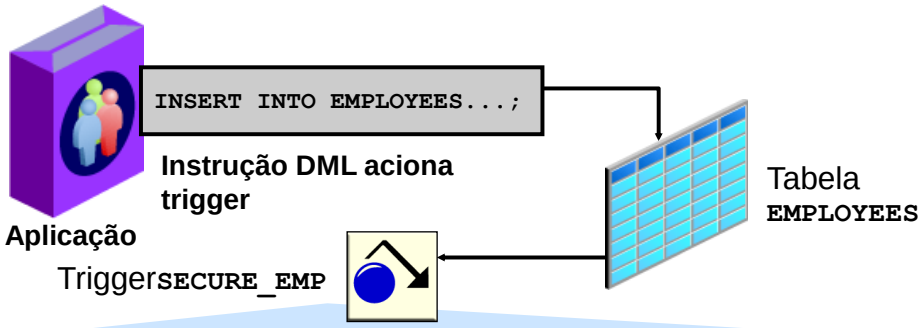
Sequência de Acionamento de Triggers: Manipulação de Várias Linhas

Quando a instrução DML que aciona o trigger afeta várias linhas, o trigger de instrução é disparado exatamente uma vez, e o de linha é disparado uma vez para cada linha afetada pela instrução.

Exemplo

A instrução SQL do slide faz com que um trigger no nível de linha seja disparado um número de vezes igual ao número de linhas que atendem à condição da cláusula `WHERE`, isto é, o número de funcionários que se reportam ao departamento 30.

Criando um Trigger de Instrução DML Exemplo: SECURE_EMP




```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON employees
  BEGIN
   IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
      (TO_CHAR(SYSDATE,'HH24:MI')
      NOT BETWEEN '08:00' AND '18:00') THEN
   RAISE_APPLICATION_ERROR(-20500, 'You may insert'
     ||' into EMPLOYEES table only during '
     ||' normal business hours.');
   END IF;
  END;
```

Criando um Trigger de Instrução DML

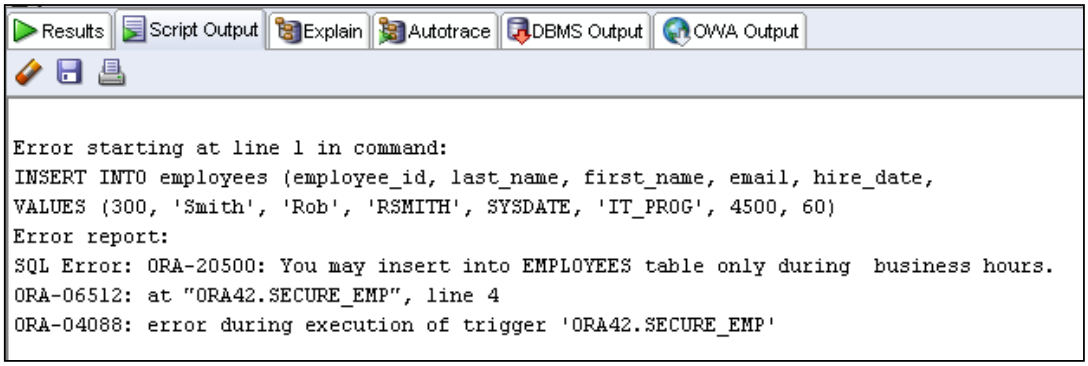
Neste exemplo, o trigger de banco de dados `SECURE_EMP` é um trigger de instrução `BEFORE` que impede que a operação `INSERT` seja bem-sucedida se determinada condição for violada.. Nesse caso, o trigger restringe as inserções na tabela `EMPLOYEES` durante determinados horários comerciais, de segunda a sexta-feira.

Se um usuário tentar inserir uma linha na tabela `EMPLOYEES` no sábado, será exibida uma mensagem de erro, haverá falha no trigger, e a instrução responsável pelo acionamento do trigger será submetida a rollback. Lembre-se de que `RAISE_APPLICATION_ERROR` é um procedure incorporado no servidor que retorna um erro para o usuário e gera uma falha no bloco PL/SQL.

Quando há falha em um trigger de banco de dados, a instrução que aciona o trigger é submetida automaticamente a rollback pelo servidor Oracle.

Testando o Trigger SECURE_EMP

```
INSERT INTO employees (employee_id, last_name,
    first_name, email, hire_date, job_id, salary,
    department_id)
VALUES (300, 'Smith', 'Rob', 'RSMITH', SYSDATE,
  'IT_PROG', 4500, 60);
```



Testando SECURE_EMP

Para testar o trigger, insira uma linha na tabela EMPLOYEES durante horários não comerciais. Quando a data e a hora estiverem fora dos horários comerciais especificados no trigger, será exibida uma mensagem de erro conforme mostrado no slide.

Usando Predicados Condicionais

```
CREATE OR REPLACE TRIGGER secure_emp BEFORE
INSERT OR UPDATE OR DELETE ON employees
  BEGIN
    IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
       (TO_CHAR(SYSDATE,'HH24')
        NOT BETWEEN '08' AND '18') THEN
     IF DELETING THEN RAISE_APPLICATION_ERROR(
        -20502,'You may delete from EMPLOYEES table'||
        'only during normal business hours.');
     ELSIF INSERTING THEN RAISE_APPLICATION_ERROR(
        -20500,'You may insert into EMPLOYEES table'||
        'only during normal business hours.');
     ELSIF UPDATING('SALARY') THEN
        RAISE_APPLICATION_ERROR(-20503, 'You may '||
        'update SALARY only normal during business hours.');
      ELSE RAISE_APPLICATION_ERROR(-20504,'You may'||
        ' update EMPLOYEES table only during'||
        ' normal business hours.');
      END IF;
    END IF;
  END;
```

Detectando a Operação DML que Acionou um Trigger

Se mais de um tipo de operação DML puder acionar um trigger (por exemplo, `ON INSERT OR DELETE OR UPDATE OF Emp_tab`), o trigger body pode usar os predicados condicionais `INSERTING`, `DELETING`, e `UPDATING` para verificar qual tipo de instrução acionou o trigger.

Você pode combinar vários eventos de trigger em um utilizando os predicados condicionais especiais `INSERTING`, `UPDATING` e `DELETING` no trigger body.

Exemplo

Crie um trigger para restringir todos os eventos de manipulação de dados na tabela `EMPLOYEES` a determinados horários comerciais, de segunda a sexta-feira, de 8:00 às 18:00.

Criando um Trigger de Linha DML

```
CREATE OR REPLACE TRIGGER restrict_salary
BEFORE INSERT OR UPDATE OF salary ON employees
FOR EACH ROW
BEGIN
  IF NOT (:NEW.job_id IN ('AD_PRES', 'AD_VP'))
     AND :NEW.salary > 15000 THEN
    RAISE_APPLICATION_ERROR (-20202,
      'Employee cannot earn more than $15,000.');
  END IF;
END;
```

```
UPDATE employees
SET salary = 15500
WHERE last_name = 'Russell';
```

```
Error starting at line 1 in command:
UPDATE employees
SET salary = 15500
WHERE last_name = 'Russell'
Error report:
SQL Error: ORA-20202: Employee cannot earn more than $15,000.
ORA-06512: at "ORA62.RESTRICT_SALARY", line 4
ORA-04088: error during execution of trigger 'ORA62.RESTRICT_SALARY'
```

Criando um Trigger de Linha DML

Você pode criar um trigger de linha `BEFORE` para impedir que a operação de trigger seja bem-sucedida se determinada condição for violada.

.No primeiro exemplo do slide, um trigger é criado para permitir que somente funcionários cujos IDs de job sejam `AD_PRES` ou `AD_VP` recebam um salário de mais de 15.000. Se você tentar atualizar o salário do funcionário `Russell` cujo ID é `SA_MAN`, o trigger gera a exceção exibida no slide.

Observação: Antes de executar o primeiro exemplo de código no slide, certifique-se de desativar os triggers `secure_emp` e `secure_employees`.

Usando os Qualificadores OLD e NEW

- Quando um trigger de nível de linha é acionado, o mecanismo de runtime PL/SQL cria e preenche duas estruturas de dados:
  - OLD: Armazena os valores originais do registro processado pelo trigger
  - NEW: Contém os novos valores
- NEW e OLD têm a mesma estrutura que um registro declarado usando %ROWTYPE na tabela à qual o trigger é associado.

Operações de Dados	Valor Antigo	Novo Valor
INSERT	NULL	Valor inserido
UPDATE	Valor antes da atualização	Valor após a atualização
DELETE	Valor antes da deleção	NULL

Usando Qualificadores OLD e NEW

Em um trigger ROW, você pode fazer referência ao valor de uma coluna antes e após a alteração dos dados, incluindo os qualificadores OLD e NEWcomo prefixos.

Observação

- Os qualificadores OLD e NEW estão disponíveis somente em triggers ROW.
- Inclua o prefixo dois-pontos (:) antes desses qualificadores em todas as instruções SQL e PL/SQL.
- O prefixo dois-pontos (:) não será usado se os qualificadores forem referenciados na condição restritiva WHEN.
- Os triggers de linha poderão prejudicar o desempenho caso sejam feitas muitas atualizações em tabelas maiores.

Usando Qualificadores OLD e NEW: Exemplo

```
CREATE TABLE audit_emp (
  user_name     VARCHAR2(30),
  time_stamp    date,
  id            NUMBER(6),
  old_last_name VARCHAR2(25),
  new_last_name VARCHAR2(25),
  old_title     VARCHAR2(10),
  new_title     VARCHAR2(10),
  old_salary    NUMBER(8,2),
  new_salary    NUMBER(8,2) )
/
CREATE OR REPLACE TRIGGER audit_emp_values
AFTER DELETE OR INSERT OR UPDATE ON employees
FOR EACH ROW
BEGIN
  INSERT INTO audit_emp(user_name, time_stamp, id,
    old_last_name, new_last_name, old_title,
    new_title, old_salary, new salary)
  VALUES (USER, SYSDATE, :OLD.employee_id,
    :OLD.last_name, :NEW.last_name, :OLD.job_id,
    :NEW.job id, :OLD.salary, :NEW.salary);
END;
```

Usando Qualificadores OLD e NEW: Exemplo

No exemplo do slide, o trigger AUDIT_EMP_VALUES é criado na tabela EMPLOYEES. O trigger inclui linhas para a tabela de um usuário, AUDIT_EMP, registrando em log a atividade do usuário na tabela EMPLOYEES. O trigger registra os valores de várias colunas antes e depois das alterações nos dados, usando os qualificadores OLD e NEWcom o respectivo nome de coluna.

Usando os Qualificadores OLD e NEW: Exemplo

```
INSERT INTO employees (employee_id, last_name, job_id,
salary, email, hire_date)
VALUES (999, 'Temp emp', 'SA_REP', 6000, 'TEMPEMP',
TRUNC(SYSDATE))
/
UPDATE  employees
 SET salary = 7000, last_name = 'Smith'
 WHERE employee_id = 999
/
SELECT *
FROM audit_emp;
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

	USER_NAME	TIME_STAMP	ID	OLD_LAST_NAME	NEW_LAST_NAME	OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
1	ORA61	04-JUN-09	(null)	(null)	Temp emp	(null)	SA_REP	(null)	6000
2	ORA61	04-JUN-09	999	Temp emp	Smith	SA_REP	SA_REP	6000	7000

Usando Qualificadores OLD e NEW: Exemplo de Uso da Tabela AUDIT_EMP

Crie um trigger na tabela EMPLOYEES para adicionar linhas a uma tabela de usuários, AUDIT_EMP, registrando a atividade de determinado usuário na tabela EMPLOYEES. O trigger registra os valores de várias colunas antes e depois das alterações nos dados, usando os qualificadores OLD e NEW com o respectivo nome de coluna.

A seguir, temos o resultado da inserção do registro do funcionário na tabela EMPLOYEES:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
999	(null)	Smith	TEMPEMP	(null)	04-JUN-09	SA_REP	7000	(null)	(null)	(null)
300	Rob	Smith	RSMITH	(null)	04-JUN-09	IT_PROG	4500	(null)	(null)	60
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACC...	8300	(null)	205	110

. . .

A seguir, este é o resultado da atualização do salário do funcionário “Smith”:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_...	MANAGER_ID	DEPARTMENT_ID
999	(null)	Smith	TEMPEMP	(null)	04-JUN-09	SA_REP	7000	(null)	(null)	(null)

. . .

Usando a Cláusula WHEN para Acionar um Trigger de Linha com Base em uma Condição

```
CREATE OR REPLACE TRIGGER derive_commission_pct
BEFORE INSERT OR UPDATE OF salary ON employees
FOR EACH ROW
WHEN (NEW.job_id = 'SA_REP')
BEGIN
 IF INSERTING THEN
   :NEW.commission_pct := 0;
 ELSIF :OLD.commission_pct IS NULL THEN
   :NEW.commission_pct := 0;
 ELSE
   :NEW.commission_pct := :OLD.commission_pct+0.05;
 END IF;
END;
/
```

Restringindo um Trigger de Linha: Exemplo

Opcionalmente, você pode incluir uma restrição de trigger na definição de um trigger de linha, especificando uma expressão Boolean SQL em uma cláusula `WHEN`. Se você incluir uma cláusula `WHEN` no trigger, então a expressão na cláusula `WHEN` é avaliada para cada linha que o trigger afeta.

Se a expressão for avaliada como `TRUE` em uma linha, então o trigger body é executado nessa linha. Entretanto, se a expressão for avaliada como `FALSE` ou `NOT TRUE` em uma linha (desconhecida, como ocorre com os nulos), então o trigger body não é executado nessa linha. A avaliação da cláusula `WHEN` não causa nenhum efeito na execução da instrução SQL de trigger (em outras palavras, a instrução que aciona o trigger não é submetida a rollback se a expressão em uma cláusula `WHEN` for avaliada como `FALSE`).

Observação: Uma cláusula `WHEN` não pode ser incluída na definição de um trigger de instrução.

No exemplo do slide, um trigger é criado na tabela `EMPLOYEES` para calcular a comissão de um funcionário quando uma linha for adicionada à tabela `EMPLOYEES`, ou quando o salário de um funcionário for modificado.

O qualificador `NEW` não pode ter o prefixo dois-pontos na cláusula `WHEN` porque a cláusula `WHEN` está fora dos blocos PL/SQL.

Resumo do Modelo de Execução de Triggers

1. Execute todos os triggers `BEFORE STATEMENT`.
2. Execute um loop *para cada linha* afetada pela instrução SQL:
   a. Execute todos os triggers `BEFORE ROW` *dessa linha*.
   b. Execute a instrução DML e a verificação da constraint de integridade *nessa linha*.
   c. Execute todos os triggers `AFTER ROW` *dessa linha*.
3. Execute todos os triggers `AFTER STATEMENT`.

Modelo de Execução de Triggers

Uma única instrução DML pode disparar até quatro tipos de triggers:

- Triggers de instrução`BEFORE` e `AFTER`
- Triggers de linha`BEFORE` e `AFTER`

Um evento de trigger ou uma instrução contida no trigger pode fazer com que uma ou mais constraints de integridade sejam verificadas. Entretanto, é possível adiar a verificação de constraints até que uma operação `COMMIT` seja executada.

Os triggers também podem fazer com que outros triggers—conhecidos como triggers em cascata—sejam acionados.

Todas as ações e verificações executadas como resultado de uma instrução SQL devem ser bem-sucedidas. Se uma exceção for gerada em um trigger e não for tratada explicitamente, todas as ações executadas como resultado da instrução SQL original serão submetidas a rollback (incluindo as executadas pelo acionamento do trigger). Isso garante que as constraints de integridade nunca sejam comprometidas por triggers.

Quando um trigger é disparado, as tabelas referenciadas na ação correspondente podem ser alteradas pelas transações de outros usuários. Em todos os casos, uma imagem com leitura consistente é garantida para os valores modificados que o trigger precisará ler (consultar) ou gravar (atualizar).

Observação: A verificação de integridade pode ser adiada até que a operação `COMMIT` seja executada.

Implementando uma Constraint de Integridade com um Trigger After

```
-- Integrity constraint violation error -2991 raised.
UPDATE employees SET department_id = 999
 WHERE employee_id = 170;
```

```
CREATE OR REPLACE TRIGGER employee_dept_fk_trg
AFTER UPDATE OF department_id ON employees
FOR EACH ROW
BEGIN
 INSERT INTO departments VALUES(:new.department_id,
              'Dept '||:new.department_id, NULL, NULL);
EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
    NULL; -- mask exception if department exists
END;
/
```

```
-- Successful after trigger is fired
UPDATE employees SET department_id = 999
 WHERE employee_id = 170;
```

```
1 rows updated
```

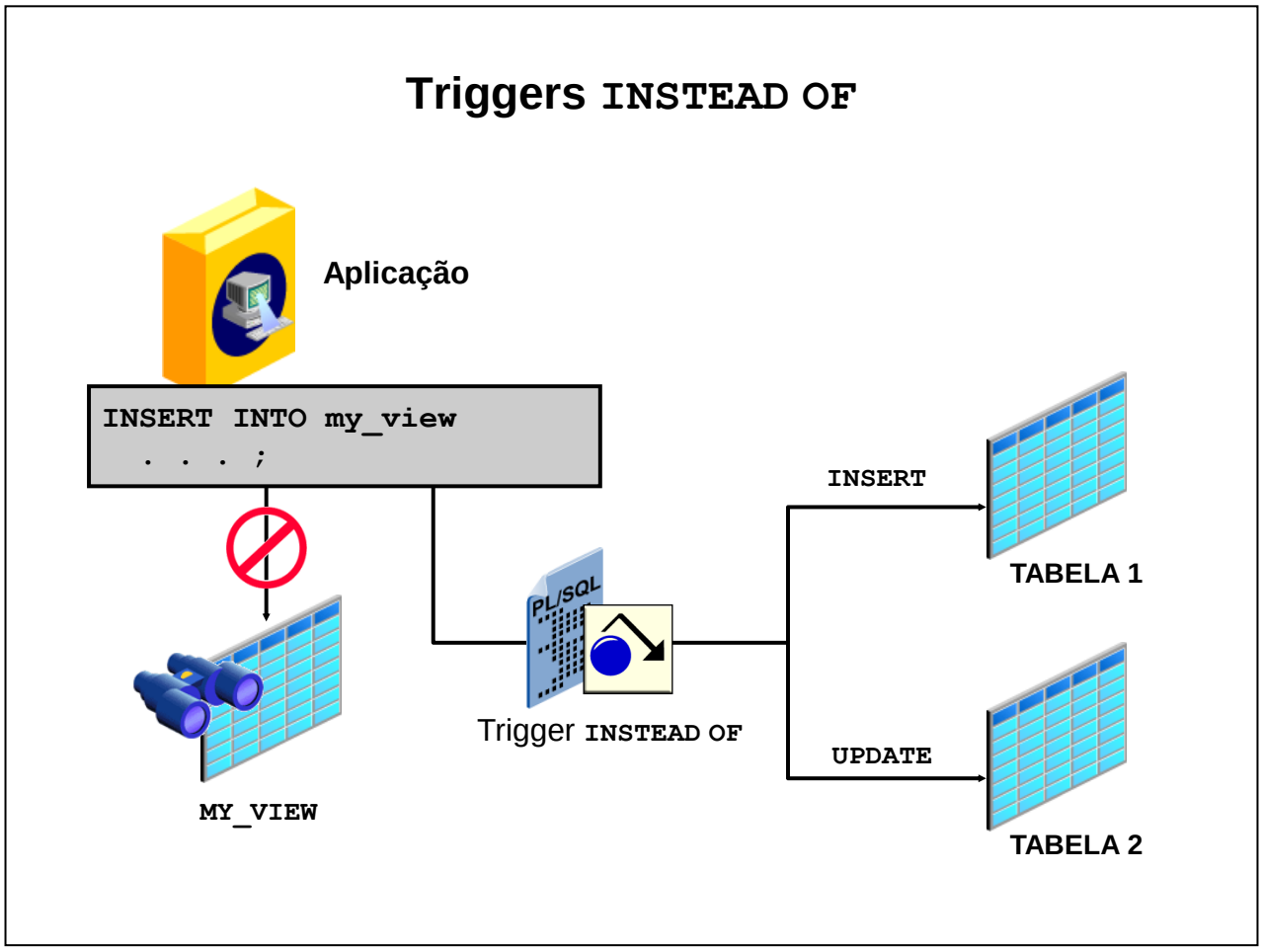
Implementando uma Constraint de Integridade com um Trigger After

O exemplo do slide explica uma situação em que a constraint de integridade pode ser implementada com um trigger AFTER. A tabela EMPLOYEES tem uma constraint de chave estrangeira na coluna DEPARTMENT_ID da tabela DEPARTMENTS.

Na primeira instrução SQL, o DEPARTMENT_ID do funcionário 170 é modificado para 999. Como o departamento 999 não existe na tabela DEPARTMENTS, a instrução gera a exceção -2291 para a violação da constraint de integridade.

É criado o trigger EMPLOYEE_DEPT_FK_TRG, que insere uma nova linha na tabela DEPARTMENTS usando :NEW.DEPARTMENT_ID para o valor do novo DEPARTMENT_ID do departamento. O trigger é acionado quando a instrução UPDATE modifica o DEPARTMENT_ID do funcionário 170 a 999. Quando a constraint da chave estrangeira é verificada, é bem-sucedida porque o trigger insere o departamento 999 na tabela DEPARTMENTS. Portanto, nenhuma exceção ocorrerá, a menos que o departamento já exista quando o trigger tentar inserir a nova linha. Entretanto, o handler EXCEPTION intercepta e mascara a exceção, permitindo que a operação seja bem-sucedida.

Observação: Embora o exemplo mostrado no slide seja de alguma forma inventado devido aos dados limitados no esquema HR, o importante é que se você adiar a verificação de constraint até o commit, então terá a capacidade de desenvolver um trigger para detectar a falha da constraint e repará-la, antes da ação do commit.

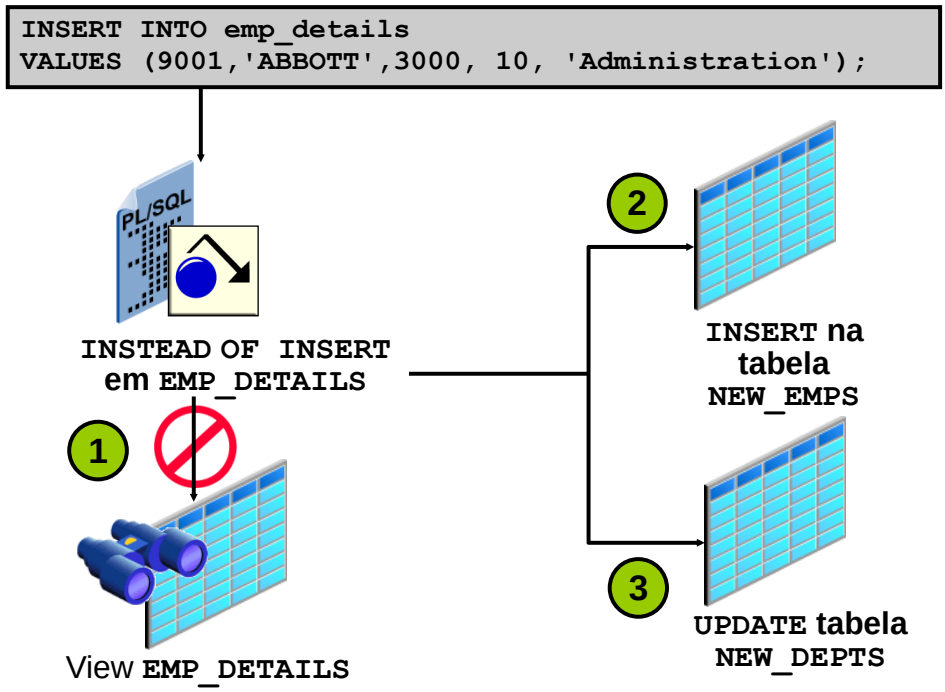


Triggers INSTEAD OF

Use os triggers INSTEAD OF para modificar dados nos quais a instrução DML foi executada para uma view inerentemente não atualizável. Esses triggers são denominados INSTEAD OF porque, ao contrário do que ocorre com outros triggers, o servidor Oracle dispara o trigger em vez de executar a instrução responsável por seu acionamento. Esses triggers são usados para executar as operações INSERT, UPDATE e DELETE diretamente nas tabelas subjacentes. Você pode criar instruções INSERT, UPDATE e DELETE em uma view, e o trigger INSTEAD OF funcionará em background, de maneira invisível, para fazer com que as ações certas sejam executadas. Não será possível modificar a view por meio de instruções DML normais se a consulta contiver operadores de conjunto, funções de grupo, cláusulas como GROUP BY, CONNECT BY, START, o operador DISTINCT ou joins. Por exemplo, se uma view consistir em mais de uma tabela, uma operação INSERT nessa view poderá ocasionar uma inserção em uma tabela e uma atualização em outra. Assim, você criará um trigger INSTEAD OF que será disparado quando ocorrer uma inserção na view. Em vez da inserção original, o trigger body será executado, o que resultará em uma inserção de dados em uma tabela e em uma atualização em outra.

Observação: Se uma view for inerentemente atualizável e tiver triggers INSTEAD OF, os triggers terão precedência. Os triggers INSTEAD OF são triggers de linha. A opção CHECK não é imposta em views quando as inserções ou as atualizações são realizadas por meio de triggers INSTEAD OF. O trigger body INSTEAD OF deve impor a verificação.

Criando um Trigger INSTEAD OF: Exemplo



Criando um Trigger INSTEAD OF

Você pode criar um trigger INSTEAD OF para manter as tabelas-base nas quais se baseiam uma view.

O exemplo ilustra a operação de inserção de um funcionário na view EMP_DETAILS, cuja consulta tem como base as tabelas EMPLOYEES e DEPARTMENTS. O trigger NEW_EMP_DEPT (INSTEAD OF) é executado no lugar da operação INSERT responsável pelo seu acionamento. Em seguida, o trigger INSTEAD OF executa as instruções INSERT e UPDATE apropriadas nas tabelas-base usadas pela view EMP_DETAILS. Portanto, em vez da inserção do registro do novo funcionário na tabela EMPLOYEES, ocorrem as seguintes ações:

1. O trigger NEW_EMP_DEPT INSTEAD OF é acionado.
2. Uma linha é inserida na tabela NEW_EMPS.
3. A coluna DEPT_SAL da tabela NEW_DEPTS é atualizada. O valor do salário fornecido para o novo funcionário é adicionado ao salário total existente do departamento ao qual o novo funcionário foi designado.

Observação: Antes de executar o exemplo no slide, você tem que criar as estruturas necessárias mostradas nas próximas duas páginas.

Criando um Trigger INSTEAD OF para Executar DML em Views Complexas

```
CREATE TABLE new_emps AS
 SELECT employee_id,last_name,salary,department_id
    FROM employees;

CREATE TABLE new_depts AS
 SELECT d.department_id,d.department_name,
        sum(e.salary) dept_sal
    FROM employees e, departments d
 WHERE  e.department_id = d.department_id;

CREATE VIEW emp_details AS
 SELECT e.employee_id, e.last_name, e.salary,
        e.department_id, d.department_name
 FROM employees e, departments d
 WHERE e.department_id = d.department_id
GROUP BY d.department_id,d.department_name;
```

Criando um Trigger INSTEAD OF (continuação)

O exemplo cria duas novas tabelas, NEW_EMPS e NEW_DEPTS, com base nas tabelas EMPLOYEES e DEPARTMENTS, respectivamente. Ele também cria uma view EMP_DETAILS a partir das tabelas EMPLOYEES e DEPARTMENTS.

Se uma view tiver uma estrutura de consulta complexa, nem sempre será possível executar o código DML diretamente nessa view de modo a afetar as tabelas subjacentes. O exemplo requer a criação de um trigger INSTEAD OF, denominado NEW_EMP_DEPT, mostrado na próxima página. O trigger NEW_DEPT_EMP trata o código DML da seguinte maneira:

- Quando uma linha é inserida na view EMP_DETAILS, em vez de serem inseridas diretamente na view, as linhas são adicionadas às tabelas NEW_EMPS e NEW_DEPTS com os valores de dados fornecidos pela instrução INSERT.
- Quando uma linha é modificada ou deletada por meio da view EMP_DETAILS, as linhas correspondentes das tabelas NEW_EMPS e NEW_DEPTS são afetadas.

Observação: Os triggers INSTEAD OF podem ser criados somente para views, e as opções BEFORE e AFTER, que definem o momento de acionamento do trigger, não são válidas.

Criando um Trigger INSTEAD OF (continuação)

```
CREATE OR REPLACE TRIGGER new_emp_dept
INSTEAD OF INSERT OR UPDATE OR DELETE ON emp_details
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO new_emps
    VALUES (:NEW.employee_id, :NEW.last_name,
            :NEW.salary, :NEW.department_id);
    UPDATE new_depts
      SET dept_sal = dept_sal + :NEW.salary
      WHERE department_id = :NEW.department_id;
  ELSIF DELETING THEN
    DELETE FROM new_emps
      WHERE employee_id = :OLD.employee_id;
    UPDATE new_depts
      SET dept_sal = dept_sal - :OLD.salary
      WHERE department_id = :OLD.department_id;
  ELSIF UPDATING ('salary') THEN
    UPDATE new_emps
      SET salary = :NEW.salary
      WHERE employee_id = :OLD.employee_id;
    UPDATE new_depts
      SET dept_sal = dept_sal +
                     (:NEW.salary - :OLD.salary)
      WHERE department_id = :OLD.department_id;
  ELSIF UPDATING ('department_id') THEN
    UPDATE new_emps
      SET department_id = :NEW.department_id
      WHERE employee_id = :OLD.employee_id;
    UPDATE new_depts
      SET dept_sal = dept_sal - :OLD.salary
      WHERE department_id = :OLD.department_id;
    UPDATE new_depts
      SET dept_sal = dept_sal + :NEW.salary
      WHERE department_id = :NEW.department_id;
  END IF;
END;
/
```

```
DEPARTMENT_ID        DEPARTMENT_NAME              DEPT_SAL
-------------------- ---------------------------- --------------------
10                   Administration               7400

1 rows selected

EMPLOYEE_ID          LAST_NAME                    SALARY               DEPARTMENT_ID
-------------------- ---------------------------- -------------------- --------------------
200                  Whalen                       4400                 10
9001                 ABBOTT                       3000                 10

2 rows selected
```

O Status de um Trigger

Um trigger existe em dois modos distintos:

- Ativado: O trigger executa sua ação se uma instrução de trigger for emitida e a restrição de trigger (se houver) for avaliada como verdadeira (default).
- Desativado: O trigger não executa sua ação mesmo se uma instrução de trigger for emitida e a restrição de trigger (se houver) for avaliada como verdadeira.

Criando um Trigger Desativado

- Antes do Oracle Database 11*g*, se você criasse um trigger cujo corpo tivesse um erro de compilação PL/SQL, então a DML para a tabela fahava.
- No Oracle Database 11*g*, você pode criar um trigger desativado e depois ativá-lo somente quando souber que ele será compilado com sucesso.

```
CREATE OR REPLACE TRIGGER mytrg
  BEFORE INSERT ON mytable FOR EACH ROW
  DISABLE
BEGIN
  :New.ID := my_seq.Nextval;
. . .
END;
/
```

Criando um Trigger Desativado

Antes do Oracle Database 11*g*, se você criasse um trigger cujo corpo tivesse um erro de compilação PL/SQL, então a DML para a tabela fahava. A seguinte mensagem de erro era exibida:

```
ORA-04098: trigger 'TRG' is invalid and failed re-validation
```

No Oracle Database 11*g*, você pode criar um trigger desativado e depois ativá-lo somente quando souber que ele será compilado com sucesso.

Também pode desativar temporariamente um trigger nas seguintes situações:

- Um objeto ao qual ele faz referência não está disponível.
- Você precisa executar uma grande carga de dados e deseja que ela proceda rapidamente sem acionar triggers.
- Você está recarregando dados.

Observação: O exemplo do código no slide supõe que você tem uma sequência existente denominada `my_seq`.

Gerenciando Triggers Usando as Instruções SQL ALTER e DROP

```
-- Desative ou reative um trigger de banco de dados:

ALTER TRIGGER trigger_name DISABLE | ENABLE;
```

```
-- Desative ou reative todos os triggers em uma tabela:

ALTER TABLE table_name DISABLE | ENABLE ALL TRIGGERS;
```

```
-- Recompile um trigger em uma tabela:

ALTER TRIGGER trigger_name COMPILE;
```

```
-- Remova um trigger do banco de dados:

DROP TRIGGER trigger_name;
```

Gerenciando Triggers

Os triggers apresentam dois modos ou estados: `ENABLED` e `DISABLED`. Quando um trigger é inicialmente criado, ele é ativado por default. O servidor Oracle verifica as constraints de integridade dos triggers ativados e garante que eles não as comprometam. Além disso, o servidor Oracle fornece views com leitura consistente para consultas e constraints, gerencia as dependências e oferece um processo de commit em duas fases quando um trigger atualiza tabelas remotas em um banco de dados distribuído.

Desativando um Trigger

Use o comando `ALTER TRIGGER` para desativar um trigger. Você também pode desativar todos os triggers em uma tabela usando o comando `ALTER TABLE`. Para melhorar o desempenho ou evitar verificações de integridade de dados durante a carga de grandes volumes de dados com utilitários como o SQL*Loader, você pode desativar triggers. Considere desativar um trigger quando ele fizer referência a um objeto de banco de dados que não esteja disponível no momento por causa de uma falha na conexão de rede ou no disco, bem como por causa de um arquivo de dados ou de um tablespace off-line.

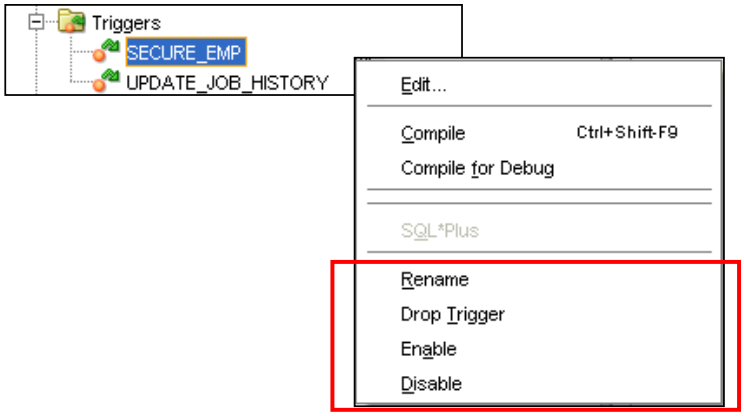
Recompilando um Trigger

Use o comando `ALTER TRIGGER` para recompilar explicitamente um trigger inválido.

Removendo Triggers

Quando um trigger não for mais necessário, use uma instrução SQL no SQL Developer ou SQL*Plus para removê -lo. Ao remover uma tabela, todos os triggers nessa tabela também são removidos.

Gerenciando Triggers Usando o SQL Developer



Gerenciando Triggers Usando o SQL Developer

Você pode usar o nó Triggers na árvore de navegação Connections para gerenciar triggers. Clique com o botão direito do mouse no nome de um trigger e selecione uma das seguintes opções:

- Edit
- Compile
- Compile for Debug
- Rename
- Drop Trigger
- Enable
- Disable

Testando Triggers

- Teste cada operação de dados que aciona um trigger, bem como as que não acionam triggers.
- Teste cada caso da cláusula `WHEN`.
- Dispare o trigger diretamente em uma operação de dados básica, bem como indiretamente em um procedure.
- Teste o efeito do trigger em outros triggers.
- Teste o efeito de outros triggers no trigger.

Testando Triggers

O teste de um código pode ser um processo demorado. Ao testar triggers, faça o seguinte:

- Certifique-se de que ele funcione de forma adequada testando vários casos separadamente:
  - Teste primeiro os cenários bem-sucedidos mais comuns.
  - Teste as condições de falha mais comuns para verificar se elas estão sendo gerenciadas corretamente.
- Quanto mais complexo for o trigger, mais detalhado deverá ser o seu teste. Por exemplo, se houver um trigger de linha com uma cláusula `WHEN` especificada, você deverá garantir que ele seja disparado quando as condições forem atendidas. Ou, se houver triggers em cascata, será necessário testar o efeito de um trigger no outro para assegurar que os resultados desejados sejam alcançados.
- Utilize o pacote `DBMS_OUTPUT` para depurar triggers.

Exibindo Informações sobre Triggers

Você pode exibir as seguintes informações sobre os triggers:

View do Dicionário de Dados	Descrição
USER_OBJECTS	Exibe informações do objeto
USER/ALL/DBA_TRIGGERS	Exibe informações sobre o trigger
USER_ERRORS	Exibe erros da sintaxe PL/SQL para um trigger

Exibindo Informações sobre Triggers

O slide mostra as views de dicionário de dados que você pode acessar para obter informações sobre os triggers.

A view `USER_OBJECTS` contém o nome e o status do trigger, bem como a data e o horário em que ele foi criado.

A view `USER_ERRORS` contém os detalhes dos erros de compilação ocorridos durante a compilação de um trigger. O conteúdo dessas views é semelhante ao de subprogramas.

A view `USER_TRIGGERS` contém detalhes, como nome, tipo, evento de trigger, tabela em que o trigger foi criado e o body do trigger.

A instrução `SELECT Username FROM USER_USERS;` fornece o nome do proprietário do trigger, e não o nome do usuário que está atualizando a tabela.

Usando USER_TRIGGERS

```
DESCRIBE user_triggers
```

```
DESCRIBE user_triggers
Name                           Null     Type
------------------------------ -------- ----------------------------------------------------
TRIGGER_NAME                            VARCHAR2(30)
TRIGGER_TYPE                            VARCHAR2(16)
TRIGGERING_EVENT                        VARCHAR2(227)
TABLE_OWNER                             VARCHAR2(30)
BASE_OBJECT_TYPE                        VARCHAR2(16)
TABLE_NAME                              VARCHAR2(30)
COLUMN_NAME                             VARCHAR2(4000)
REFERENCING_NAMES                       VARCHAR2(128)
WHEN_CLAUSE                             VARCHAR2(4000)
STATUS                                  VARCHAR2(8)
DESCRIPTION                             VARCHAR2(4000)
ACTION_TYPE                             VARCHAR2(11)
TRIGGER_BODY                            LONG()
CROSSEDITION                            VARCHAR2(7)
BEFORE_STATEMENT                        VARCHAR2(3)
BEFORE_ROW                              VARCHAR2(3)
AFTER_ROW                               VARCHAR2(3)
AFTER_STATEMENT                         VARCHAR2(3)
INSTEAD_OF_ROW                          VARCHAR2(3)
FIRE_ONCE                               VARCHAR2(3)
APPLY_SERVER_ONLY                       VARCHAR2(3)

21 rows selected
```

```
SELECT trigger_type, trigger_body
FROM   user_triggers
WHERE trigger_name = 'SECURE_EMP';
```

Usando USER_TRIGGERS

Se o arquivo-fonte não estiver disponível, você poderá usar o SQL Worksheet no SQL Developer ou SQL*Plus para gerá-lo novamente em USER_TRIGGERS. Você também pode examinar as views ALL_TRIGGERS e DBA_TRIGGERS, que contêm a coluna adicional OWNER, referente ao proprietário do objeto. A saída do segundo exemplo do slide é a seguinte:

```
TRIGGER_TYPE     TRIGGER_BODY
---------------- --------------------------------------------------------------------------------
BEFORE STATEMENT BEGIN
                  IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
                     (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN '08' AND '18') THEN
                    IF DELETING THEN
                      RAISE_APPLICATION_ERROR(-20502,
                        'You may delete from EMPLOYEES table only during normal business hours.');
                    ELSIF INSERTING THEN
                      RAISE_APPLICATION_ERROR(-20500,
                        'You may insert into EMPLOYEES table only during normal business hours.');
                    ELSIF UPDATING('SALARY') THEN
                      RAISE_APPLICATION_ERROR(-20503,
                        'You may update SALARY only during normal business hours.');
                    ELSE
                      RAISE_APPLICATION_ERROR(-20504,
                        'You may update EMPLOYEES table only during normal business hours.');
                    END IF;
                  END IF;
                 END;
```

Questionário

Um evento de trigger pode ser um ou mais dos seguintes:

1. Uma instrução `INSERT`, `UPDATE`, ou `DELETE` em uma tabela específica (ou view, em alguns casos)
2. Uma instrução `CREATE`, `ALTER`, ou `DROP` em qualquer objeto de esquema
3. Um shutdown de instância ou inicialização de banco de dados
4. Uma mensagem de erro específica ou qualquer mensagem de erro
5. Um logon ou logoff de usuário

Resposta: 1, 2, 3, 4, 5

Sumário

Nesta lição, você aprendeu a:

- Criar triggers de banco de dados chamados por operações DML
- Criar triggers de instrução e de linha
- Usar regras de acionamento de triggers de banco de dados
- Ativar, desativar e gerenciar triggers de banco de dados
- Desenvolver uma estratégia para o teste de triggers
- Remover triggers de banco de dados

Sumário

Esta lição abordou a criação de triggers de banco de dados que são executados antes, depois ou em vez de uma operação DML especificada. Os triggers são associados a tabelas ou views de banco de dados. As opções `BEFORE` e `AFTER`, que definem o momento de acionamento do trigger, aplicam-se a operações DML em tabelas. O trigger `INSTEAD OF` é usado para substituir operações DML em uma view pelas instruções DML adequadas em outras tabelas do banco de dados.

Embora os triggers sejam ativados por default, eles podem ser desativados para que a operação correspondente seja suprimida até que eles sejam reativados. Se as regras de negócios mudarem, os triggers poderão ser removidos ou alterados conforme necessário.

Visão Geral do Exercício 12: Criando Triggers de Instrução e de Linha

Este exercício aborda os seguintes tópicos:

- Criando triggers de linha
- Criando um trigger de instrução
- Chamando procedures em um trigger

Exercício 12: Visão Geral

Neste exercício, você criará triggers de instrução e de linha, além de procedures chamados a partir dos triggers.