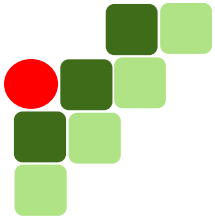
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Araraquara

Banco de Dados II

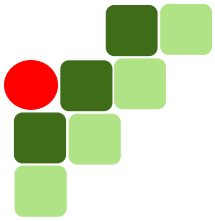
Cristiane Yaguinuma
cristiane.yaguinuma@ifsp.edu.br

- Triggers (gatilhos)



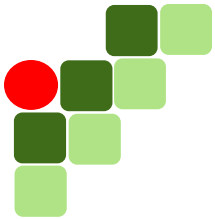
Roteiro da aula

- Triggers
  - Definição
  - Componentes
  - CREATE TRIGGER
  - Comandos para triggers
  - Recomendações
  - Exercícios



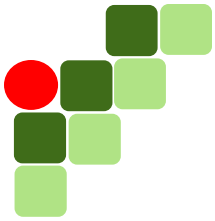
Triggers

- Procedimentos armazenados no banco de dados
  - Objetos do banco de dados
- Disparados implicitamente quando um evento acontece
  - Um comando DML (DELETE, INSERT ou UPDATE)
    - DML Trigger
  - Um comando DDL (CREATE, ALTER ou DROP)
  - Uma operação de BD (SERVERERROR, LOGON, LOGOFF, STARTUP ou SHUTDOWN)
    - System Trigger



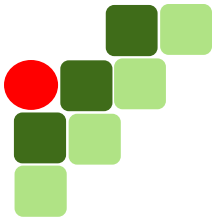
Por que utilizar triggers?

- Atualizar dados derivados automaticamente
- Monitorar o BD
  - Acompanhar o que ocorre após eventos no BD
  - Acessos que alterem linhas de uma tabela ou eventos que ocorram no BD podem ser registrados em outra tabela (auditoria)
- Manter a consistência do BD
  - Modificar dados de tabelas quando comandos DML forem executados
  - Garantir regras de negócio complexas



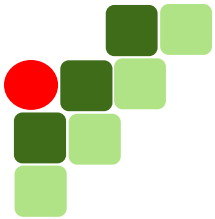
Triggers

- Parte do padrão ANSI/SQL-99
- Disponíveis na maioria dos SGBDs relacionais
- No Oracle, podem ser implementados em:
  - Java
  - PL/SQL
- PL/SQL é usada na maioria das vezes



Componentes de um trigger

- Evento(s)
  - Dispara(m) automaticamente a execução do trigger
  - Em geral, comandos DML
- Condição
  - Determina quando a ação do trigger deve ser executada
- Ação
  - Sequência de instruções SQL ou PL/SQL a serem executadas



Configurando o ambiente

- Criar as tabelas:

Departamentos (codigo, nome, *total_func*)

`total_func` tem valor default 0

codigo e nome obrigatórios

Funcionarios (codigo, nome, data_contr, salario, cod_depto)

cod_depto referencia Departamentos(codigo)

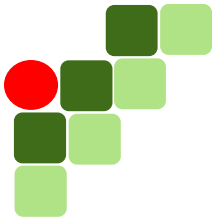
codigo e nome obrigatórios

- Autorizar usuário a criar triggers
  - GRANT CREATE TRIGGER

Criando um trigger

```
CREATE OR REPLACE TRIGGER DML_FUNC
  BEFORE INSERT ON FUNCIONARIOS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Novo funcionário inserido.');
END;
/
```

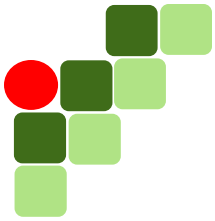
```
SET SERVEROUTPUT ON
INSERT INTO FUNCIONARIOS(CODIGO, NOME)
VALUES (1, 'CRISTIANE');

INSERT INTO FUNCIONARIOS(CODIGO, NOME)
VALUES (2, 'FERNANDO');
```



Criando um trigger

```
CREATE OR REPLACE TRIGGER DML_FUNC
  BEFORE INSERT OR UPDATE OR DELETE ON FUNCIONARIOS
BEGIN
  CASE
    WHEN INSERTING THEN
      DBMS_OUTPUT.PUT_LINE('Novo funcionário inserido');
    WHEN UPDATING THEN
      DBMS_OUTPUT.PUT_LINE('Funcionário atualizado.');
    WHEN DELETING THEN
      DBMS_OUTPUT.PUT_LINE('Funcionário removido.');
  END CASE;
END;
/
```

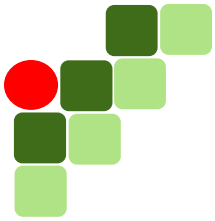
```
CREATE OR REPLACE TRIGGER DML_FUNC
  BEFORE INSERT OR
         UPDATE OF DATA_CONTR, SALARIO OR
         DELETE
  ON FUNCIONARIOS
BEGIN
  CASE
    WHEN INSERTING THEN
      DBMS_OUTPUT.PUT_LINE('Novo funcionário inserido');
    WHEN UPDATING('DATA_CONTR') THEN
      DBMS_OUTPUT.PUT_LINE('Funcionário atualizado: data
contratação');
    WHEN UPDATING('SALARIO') THEN
      DBMS_OUTPUT.PUT_LINE('Funcionário atualizado:
salario');
    WHEN DELETING THEN
      DBMS_OUTPUT.PUT_LINE('Funcionário removido.');
  END CASE;
END;
/
```



Statement trigger

- Executa apenas uma vez para o comando DML que o disparou, mesmo que afete várias tuplas

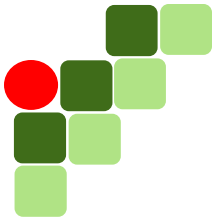
```
UPDATE FUNCIONARIOS
SET SALARIO = 11000;

UPDATE FUNCIONARIOS
SET DATA_CONTR = SYSDATE;
```



Row-level trigger

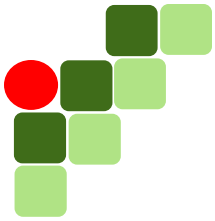
- É executado para cada tupla afetada pelo comando DML que o disparou
- Adicionar a opção FOR EACH ROW

```
CREATE OR REPLACE TRIGGER DML_FUNC
  BEFORE INSERT OR
         UPDATE OF DATA_CONTR, SALARIO OR
         DELETE
  ON FUNCIONARIOS
  FOR EACH ROW
BEGIN
...
END;
/
```



Pseudoregistros

- Como acessar os valores das tuplas afetadas por um row-level trigger?
  - Quais os valores inseridos em um INSERT?
  - Quais os valores da tupla antes e depois de um UPDATE?
  - Quais os valores antes de um DELETE?
- Usar pseudoregistros :OLD e :NEW
  - :OLD → estado da tupla antes do comando do trigger ser executado
  - :NEW → estado da tupla depois do comando do trigger ser executado



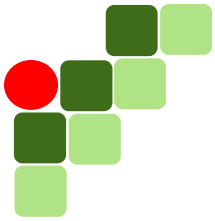
Pseudoregistros

- Usados para acessar os valores das tuplas modificadas
  - :OLD.SALARIO
  - :NEW.SALARIO

Triggering Statement	OLD.*field* Value	NEW.*field* Value
INSERT	NULL	Post-insert value
UPDATE	Pre-update value	Post-update value
DELETE	Pre-delete value	NULL

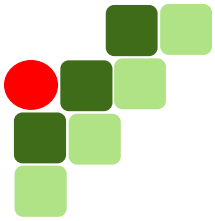
Table 9-1

```
CREATE OR REPLACE TRIGGER DML_FUNC
  BEFORE INSERT OR UPDATE OR DELETE
  ON FUNCIONARIOS
  FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('DADOS ANTIGOS:' || chr(10) ||
  'CODIGO = ' || :OLD.CODIGO || chr(10) ||
  'NOME = ' || :OLD.NOME || chr(10) ||
  'DATA CONTRATACAO = ' || :OLD.DATA_CONTR || chr(10) ||
  'SALARIO = ' || :OLD.SALARIO || chr(10) ||
  'COD_DEPTO = ' || :OLD.COD_DEPTO || chr(10));

  DBMS_OUTPUT.PUT_LINE('DADOS NOVOS:' || chr(10) ||
  'CODIGO = ' || :NEW.CODIGO || chr(10) ||
  'NOME = ' || :NEW.NOME || chr(10) ||
  'DATA CONTRATACAO = ' || :NEW.DATA_CONTR || chr(10) ||
  'SALARIO = ' || :NEW.SALARIO || chr(10) ||
  'COD_DEPTO = ' || :NEW.COD_DEPTO || chr(10));
END;
/
```



Exercício

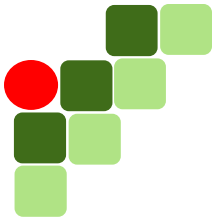
- Definir o trigger imprimir_alteracao_salario que imprima na tela as alterações de salário de funcionários
  - Salário antigo
  - Salário novo
  - Diferença entre os valores de salário



Exercício

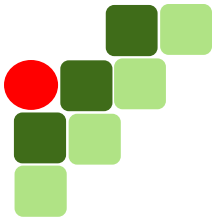
```
CREATE OR REPLACE TRIGGER imprimir_alteracao_salario
  BEFORE INSERT OR UPDATE OF salario ON funcionarios
  FOR EACH ROW
DECLARE
    diferenca number;
BEGIN
    diferenca := :NEW.salario - :OLD.salario;
    dbms_output.put_line('salario antigo: ' || :OLD.salario);
    dbms_output.put_line('salario novo: ' || :NEW.salario);
    dbms_output.put_line('diferença: ' || diferenca);
END;
/
```

Bloco de código PL/SQL



Cláusula WHEN

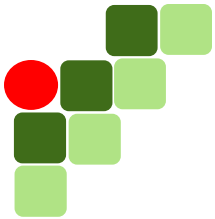
```
CREATE OR REPLACE TRIGGER imprimir_alteracao_salario
  BEFORE INSERT OR UPDATE OF salario ON emp.funcionarios
  FOR EACH ROW
WHEN (NEW.salario > 10000)
DECLARE
    diferenca number;
BEGIN
    diferenca := :NEW.salario - :OLD.salario;
    dbms_output.put_line('salario antigo: ' || :OLD.salario);
    dbms_output.put_line('salario novo: ' || :NEW.salario);
    dbms_output.put_line('diferença: ' || diferenca);
END;
/
```



Triggers BEFORE e AFTER

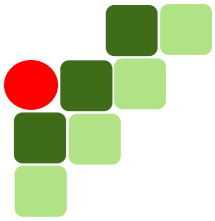
- BEFORE
  - Usado para modificar os valores das tuplas antes de serem escritos em disco
  - Modificações em :NEW serão efetivadas em disco

- AFTER
  - Utiliza valores que já foram escritos em disco
  - Modificação de valores de :NEW não fazem efeito



Exercício

- Definir o trigger regra_salario que impeça alteração de salário para valores acima de 20000 e imprima na tela um aviso sobre essa tentativa de alteração

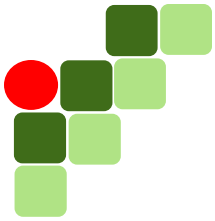


Comandos para triggers

- Para confirmar a criação do Trigger:
  - `select trigger_name from user_triggers;`
- Para eliminar:
  - `drop trigger` ***`nome_trigger`***`;`
- Para desabilitar/habilitar:

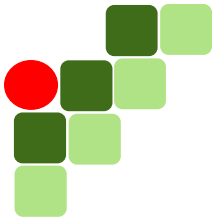
```
alter table nome_tabela disable | enable all
triggers;
```

```
alter trigger nome_trigger disable | enable;
```



Recomendações

- Não definir triggers que duplicam funcionalidades existentes no SGBD
  - Validação: melhor usar *constraint* do tipo CHECK
- Limitar o tamanho
  - Caso um trigger seja muito extenso, é melhor criar um procedimento armazenado e chamá-lo a partir do trigger



Recomendações

- Não criar triggers recursivos

```
CREATE TRIGGER trigger1
    AFTER INSERT ON tabela1
    FOR EACH ROW
       UPDATE tabela2 SET ...

CREATE TRIGGER trigger2
    AFTER UPDATE ON tabela2
    FOR EACH ROW
       INSERT INTO tabela1 VALUES ...
```

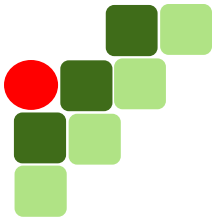


Tabela Mutante

- Tabela mutante: tabela que está sendo modificada por um comando DML
- A restrição de tabela mutante evita que o trigger faça consultas ou modifique a tabela que está sendo modificada pelo evento que o disparou
- Ocorre somente para row-level trigger
- Erro ORA-04091 é disparado e os efeitos do trigger e do evento que o disparou são desfeitos (ROLLBACK)

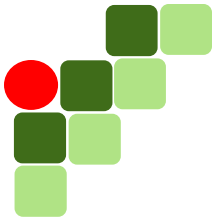


Tabela Mutante

- Regra de salário: aumento de salário somente é permitido se o novo salário for menor ou igual a média de salário de todos os funcionários

```
CREATE OR REPLACE TRIGGER REGRA_SALARIO
BEFORE UPDATE OF SALARIO ON FUNCIONARIOS
FOR EACH ROW
DECLARE
  AVG_SAL NUMBER;
BEGIN
  SELECT AVG(SALARIO) INTO AVG_SAL FROM FUNCIONARIOS;
  IF :NEW.SALARIO > AVG_SAL THEN
    Raise_Application_Error(-20000, 'AUMENTO ACIMA DA
MÉDIA DE SALÁRIO');
  END IF;
END;
/
```

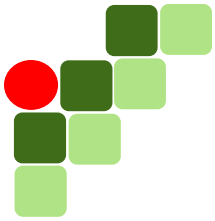
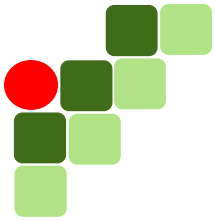


Tabela Mutante

- Para evitar o erro de tabela mutante
  - COMPOUND TRIGGER

```
CREATE OR REPLACE TRIGGER REGRA_SALARIO
FOR UPDATE OF SALARIO ON EMP.FUNCIONARIOS
COMPOUND TRIGGER
  AVG_SAL NUMBER;
  BEFORE STATEMENT IS
  BEGIN
    SELECT AVG(SALARIO) INTO AVG_SAL
    FROM FUNCIONARIOS;
  END BEFORE STATEMENT;

  BEFORE EACH ROW IS
  BEGIN
    IF :NEW.SALARIO > AVG_SAL THEN
      Raise_Application_Error(-20000, 'NÃO É PERMITIDO
AUMENTO ACIMA DA MÉDIA DE SALÁRIO');
    END IF;
  END BEFORE EACH ROW;
END;
/
```



Referências

- https://docs.oracle.com/cd/E11882_01/appdev.112/e25519/triggers.htm
- https://docs.oracle.com/cloud/latest/db112/LNPLS/create_trigger.htm
- https://docs.oracle.com/database/121/TDDDG/tdddg_triggers.htm