FIAP

Autonomous Transactions e Packages

DATABASE APPLICATION DEVELOPMENT

Leonardo Uchida
leonardo@fiap.com.br

Autonomous Transaction

FIAP

Transação autônoma é o mecanismo em PL/SQL para realizar um processamento independente da sessão que originou a execução. Permite subdividir transações e múltiplos controles de commit e rollback.

- ✓ Garante independência de controle de efetividade
- ✓ Permite efetivar transações secundárias sem influenciar na sessão principal
- ✓ Diretiva pré-compilada, pois necessita ser executada em sessão independente
- ✓ Deve ser utilizada com cautela, pois pode ocasionar locks e deadlocks

Autonomous Transaction

FIAP

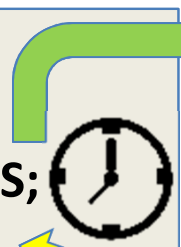
Sintaxe:

```
CREATE OR REPLACE PROCEDURE exemplo_autonomous
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
        INSERT INTO LOG VALUES ….
        COMMIT;
        …
EXCEPTION
        …
END;
```

Transação Principal

```
...
DELETE FROM FUNC;
EXEMPLO_AUTONOMOUS;

ROLLBACK;
...
```



Transação Autônoma

```
PROCEDURE exemplo_autonomous
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
...
COMMIT;
```

Autonomous Transaction

FIAP

Exemplo:

```
CREATE OR REPLACE PROCEDURE log_errors (p_error_message IN VARCHAR2)
AS
   PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   INSERT INTO log_carga (id_log, log_timestamp, error_message)
    VALUES (error_logs_seq.NEXTVAL, SYSTIMESTAMP, p_error_message);
   COMMIT;
END;
```

Packages

FIAP

É um objeto que agrupa logicamente/empacota outros objetos PL/SQL com o objetivo de organizar e centralizar a gestão de códigos.

- ✓ Deve ter uma SPEC e opcionalmente uma BODY.
- ✓ Modularização por assunto
- ✓ Permite o uso de constantes globais
- ✓ Melhor performance
- ✓ Facilidade em gerir privilégios (Parcial)
- ✓ Melhor gestão de dependência com outros objetos

Packages

FIAP

Sintaxe:

```
CREATE OR REPLACE PACKAGE PKG_CADASTRO
            PROCEDURE CAD_CLIENTE (...);
            PROCEDURE CAD_FUNCIONARIO(...);
            FUNCTION RET_DEPTO(...) RETURN VARCHAR2;
END;

CREATE OR REPLACE PACKAGE BODY PKG_CADASTRO
            PROCEDURE CAD_CLIENTE (...) IS
              BEGIN
                ....
              END;

            PROCEDURE CAD_FUNCIONARIO(...) IS ... END;
            FUNCTION RET_DEPTO(...) RETURN VARCHAR2 IS ... END;
END;
```

Packages

FIAP

Exemplo:

Package Specification

```
create or replace package pck_exemplo1 is
  function fnc_soma (p_n1 number, p_n2 number) return number;
  function fnc_mult (p_n1 number, p_n2 number) return number;
  procedure prc_atualiza_salario;
  procedure prc_saque(p_cc varchar2, valor number);
  function fnc_depto (p_cd_depto loc_depto.cd_depto%type) return varchar2;
end;
```

Packages

FIAP

Exemplo:

Package Body (1/3)

```
create or replace package body pck_exemplo1 is
 function fnc_soma (p_n1 number, p_n2 number) return number is
 begin
  return p_n1 + p_n2;
 exception
  when others then
   return 0;
 end;
 function fnc_mult (p_n1 number, p_n2 number) return number is
 begin
  return p_n1*p_n2;
 exception
  when others then
   return 0;
 end;
 procedure prc_atualiza_salario is
 begin
  null;
 end;
```

Packages

FIAP

Exemplo:

Package Body (2/3)

```
procedure prc_saque(p_cc varchar2, valor number) is
  begin
    null;
  end;

  function fnc_depto
(p_cd_depto loc_depto.cd_depto%type) return varchar2
is
  cursor c_depto(p_cod loc_depto.cd_depto%type) is
    select nm_depto
      from loc_depto
     where cd_depto = p_cod;

  v_depto c_depto%rowtype;
  v_nome  loc_depto.nm_depto%type;
  erro_nao_encontrado   exception;
begin
  open c_depto (p_cd_depto);
  loop
    fetch c_depto into v_depto;
```

Packages

FIAP

Exemplo:

Package Body (3/3)

```
exit when c_depto%notfound;
     v_nome := v_depto.nm_depto;
  end loop;
  if v_nome is null then
    raise erro_nao_encontrado;
  end if;
  return v_nome;
exception
  when erro_nao_encontrado then
    raise_application_error (-20001, 'Depto '|| p_cd_depto ||'não encontrado');
  when others then
    raise_application_error (-20002, 'Erro não tratado.');
end;
end;
```