Monitoria GDI
Aula Prática

DML + PL/SQL

parte 2

DML

linguagem de manipulação de dados

Exercício 1

- **Mostre todas as notas do período de '2010.2' do aluno de nome 'Augustos Kilter'.**

Exercício 1

- **Mostre todas as notas do período de '2010.2' do aluno de nome 'Augustos Kilter'.**

```
SELECT PV.nota
FROM Pessoa P
   INNER JOIN Aluno A
      ON P.matricula_pessoa = A.matricula_aluno
   INNER JOIN Prova PV
      ON PV.matricula_aluno = A.matricula_aluno
WHERE P.nome = 'Augustus Kilter'
   AND PV.ano_semestre = '2010.2';
```

Exercício 2

- **Para o aluno de nome 'Joao Custodia' mostre todos os projetos dos quais ele já participou, ordenando-os por período e conceito obtido.**

Exercício 2

- **Para o aluno de nome 'Joao Custodia' mostre todos os projetos dos quais ele já participou, ordenando-os por período e conceito obtido.**

```
SELECT PJ.Titulo, AT.ano_semestre, PJ.conceito
FROM Pessoa P
    INNER JOIN Aluno A
       ON P.matricula_pessoa = A.matricula_aluno
    INNER JOIN Aluno_turma AT
       ON A.matricula_aluno = AT.matricula_aluno
    INNER JOIN Projeto PJ
       ON PJ.codigo_projeto = AT.codigo_projeto
WHERE P.nome = 'Joao Custodia'
ORDER BY AT.ano_semestre, PJ.conceito;
```

Exercício 3

- **Liste o nome e a matrícula dos professores que ensinaram à aluna 'Helena Nunes' no seu primeiro período. Também informe o código das disciplinas cursadas.**

```
SELECT
P2.nome,PR.matricula_professor,AT.codigo_disciplina
FROM Pessoa P
   JOIN Aluno A
      ON A.matricula_aluno = P.matricula_pessoa
   JOIN Aluno_turma AT
      ON AT.matricula_aluno = A.matricula_aluno
   JOIN Ministra M
      ON M.codigo_disciplina = AT.codigo_disciplina
         AND M.codigo_curso = AT.codigo_curso
         AND M.ano_semestre = AT.ano_semestre
   JOIN Professor PR
      ON PR.matricula_professor = M.matricula_professor
   JOIN Pessoa P2
      ON PR.matricula_professor = P2.matricula_pessoa

WHERE P.nome = 'Helena Nunes'
   AND AT.ano_semestre =
      (SELECT MIN(ano_semestre)
      FROM Aluno A2
         INNER JOIN Aluno_turma AT2
         ON AT2.matricula_aluno = A2.matricula_aluno
      WHERE A2.matricula_aluno = A.matricula_aluno);
```

Exercício 4

- **Para todos os alunos que pagaram a disciplina 5 mostre os projetos que foram desenvolvidos por eles bem como seu período de execução. Mostre título e curso dos projetos. Mesmo os alunos sem projeto deverão ser exibidos.**

Exercício 4

- **Para todos os alunos que pagaram a disciplina 5 mostre os projetos que foram desenvolvidos por eles bem como seu período de execução. Mostre título e curso dos projetos. Mesmo os alunos sem projeto deverão ser exibidos.**

```
SELECT DISTINCT P.nome, PJ.titulo, AT.ano_semestre,C.nome
FROM Pessoa P
    INNER JOIN Aluno A
       ON P.matricula_pessoa = A.matricula_aluno
    INNER JOIN Aluno_turma AT
       ON A.matricula_aluno = AT.matricula_aluno
    LEFT OUTER JOIN Projeto PJ
       ON AT.codigo_projeto = PJ.codigo_projeto
    INNER JOIN Curso C
       ON AT.codigo_curso = C.codigo_curso
WHERE AT.codigo_disciplina = 5
ORDER BY P.nome, AT.ano_semestre;
```

Exercício 5

- **Considere que todo professor é um líder em potencial. Realize uma consulta que relacione, em duas colunas, os nomes dos professores e o nome dos seus líderes. Mesmo os professores que não têm líder deverão aparecer na primeira coluna e mesmo os professores que não têm liderados devem aparecer na lista de líderes (segunda coluna).**

Exercício 5

- **Considere que todo professor é um líder em potencial. Realize uma consulta que relacione, em duas colunas, os nomes dos professores e o nome dos seus líderes. Mesmo os professores que não têm líder deverão aparecer na primeira coluna e mesmo os professores que não têm liderados devem aparecer na lista de líderes (segunda coluna).**

```
SELECT p1.nome AS liderado, p2.nome AS lider
FROM professor pr1
   FULL OUTER JOIN professor pr2
      ON (pr1.matricula_lider = pr2.matricula_professor)
   LEFT OUTER JOIN pessoa p1
      ON (p1.matricula_pessoa = pr1.matricula_professor)
   LEFT OUTER JOIN pessoa p2
      ON (p2.matricula_pessoa = pr2.matricula_professor);
```

Exercício 6

- **Exiba o código e o nome de TODOS os cursos bem como a quantidade de alunos que estão vinculados a ele, ordenando-os por essa quantidade.**

Exercício 6

- **Exiba o código e o nome de TODOS os cursos bem como a quantidade de alunos que estão vinculados a ele, ordenando-os por essa quantidade.**

```
SELECT C.codigo_curso,C.nome,COUNT(A.matricula_aluno) AS QTD
FROM Curso C
   LEFT OUTER JOIN Aluno A
      ON A.codigo_curso = C.codigo_curso
GROUP BY C.codigo_curso, C.nome
ORDER BY QTD DESC;
```

Exercício 7

- **Mostre, para cada um dos professores, a quantidade de alunos diferentes que já passaram por sua orientação.**

Exercício 7

- **Mostre, para cada um dos professores, a quantidade de alunos diferentes que já passaram por sua orientação.**

```
SELECT PR.matricula_professor, P.nome,
   COUNT(DISTINCT AT.matricula_aluno) AS QTD
FROM Pessoa P
   JOIN Professor PR
      ON P.matricula_pessoa = PR.matricula_professor
   JOIN Ministra M
      ON M.matricula_professor = PR.matricula_professor
   JOIN Aluno_turma AT
      ON AT.codigo_disciplina = M.codigo_disciplina
         AND AT.codigo_curso = M.codigo_curso
         AND AT.ano_semestre = M.ano_semestre
GROUP BY PR.matricula_professor, P.nome
ORDER BY QTD DESC;
```

Exercício 8

- **Considere que haverá um recálculo dos alunos aprovados no vestibular.**
- **Só serão aprovados aqueles que tiverem obtido uma nota no máximo 5% menor que a média das notas dos alunos daquele curso.**
- **Mostre os alunos que deveriam abandonar os cursos, o código do curso e a nota, de acordo com as novas regras.**

Exercício 8

- Considere que haverá um recálculo dos alunos aprovados no vestibular.
- Só serão aprovados aqueles que tiverem obtido uma nota no máximo 5% menor que a média das notas dos alunos daquele curso.
- Mostre os alunos que deveriam abandonar os cursos, o código do curso e a nota, de acordo com as novas regras.

```
SELECT P.nome, A.codigo_curso, a.NOTA_VESTIBULAR
FROM Pessoa P
    INNER JOIN Aluno A
        ON P.matricula_pessoa = A.matricula_aluno
WHERE A.nota_vestibular <
       (SELECT (AVG(A2.nota_vestibular)) * 0.95
       FROM Aluno A2
       WHERE A2.codigo_curso = A.codigo_curso)
ORDER BY A.codigo_curso;
```

Exercício 9

- **Utilizando a mesma ideia da consulta anterior, liste as informações dos projetos que devem ser cancelados por apresentar qualquer tipo de problema (reprovação no vestibular) com algum dos alunos envolvidos.**

Exercício 9

- **Utilizando a mesma ideia da consulta anterior, liste as informações dos projetos que devem ser cancelados por apresentar qualquer tipo de problema (reprovação no vestibular) com algum dos alunos envolvidos.**

```
SELECT DISTINCT PJ.titulo
FROM Projeto PJ
   INNER JOIN Aluno_turma AT
      ON AT.codigo_projeto = PJ.codigo_projeto
WHERE AT.matricula_aluno = ANY
         (SELECT A.matricula_aluno
         FROM Aluno A
         WHERE A.nota_vestibular <
                  (SELECT (AVG(A2.nota_vestibular)) * 0.95
                  FROM Aluno A2
                  WHERE A2.codigo_curso = A.codigo_curso));
```

Exercício 10

- **Na mesma consulta mostre os cursos com maior e pior média de notas no vestibular.**

Exercício 10

- **Na mesma consulta mostre os cursos com maior e pior média de notas no vestibular.**

```
SELECT MAX(nova_tabela.Media), MIN(nova_tabela.Media)
FROM (SELECT codigo_curso, AVG(nota_vestibular) AS Media
      FROM Aluno
      GROUP BY codigo_curso) nova_tabela;
```

Exercício 11

- **Mostre todos os professores que são líderes. (Use EXISTS)**

Exercício 11

- **Mostre todos os professores que são líderes. (Use EXISTS)**

```
SELECT P.nome
FROM Pessoa P, Professor PR
WHERE P.matricula_pessoa = PR.matricula_professor
   AND EXISTS
      (SELECT PR2.matricula_professor
      FROM Professor PR2
      WHERE PR2.matricula_lider = PR.matricula_professor);
```

PL/SQL

Procedural Language/ Structured Query Language

Exercício 12

- **Suponha que você foi contratado pelo setor financeiro da Universidade.**
  Seu trabalho será o de calcular o valor total, em dinheiro, que uma pessoa deverá receber durante o semestre escolhido.
- **No caso dos professores, para cada disciplina ministrada eles recebem R$ 800,00 por mês. Caso coordenem alguma disciplina recebem uma gratificação de R$ 137,00 por mês. Se o professor exercer alguma liderança, então ele recebe um bonificação única de R$ 106,00 por semestre.**
- **Para os alunos, cada monitoria lhes rende uma bolsa-auxílio de R$ 53,00 por mês, enquanto durar o semestre letivo (4 meses). Para cada projeto produzido durante o semestre, que tenha obtido conceito BOM, o aluno recebe uma premiação por direitos autorais de R$ 102,00.**

Exercício 12

- **Ainda, caso o aluno tenha obtido nota no vestibular igual ou superior a 8,0 ele tem direito a um bolsa de vale-transporte no valor de R$ 27,00 mensais enquanto durar seu curso. Caso sua nota tenha sido de 5,0 até 8,0 o valor da bolsa cai para R$ 15,00 mensais.**

Exercício 12

- **Ainda, caso o aluno tenha obtido nota no vestibular igual ou superior a 8,0 ele tem direito a um bolsa de vale-transporte no valor de R$ 27,00 mensais enquanto durar seu curso. Caso sua nota tenha sido de 5,0 até 8,0 o valor da bolsa cai para R$ 15,00 mensais.**

```
CREATE OR REPLACE FUNCTION calcula_salario
   (mat IN pessoa.matricula_pessoa%TYPE,
   semestre IN turma.ano_semestre%TYPE)
   RETURN NUMBER IS

   retorno NUMBER;
   qtd_prof NUMBER;
   qtd_alun NUMBER;
   prof professor.matricula_professor%TYPE;
   alun aluno.matricula_aluno%TYPE;
```

```
BEGIN
   SELECT count(*) INTO qtd_prof
   FROM professor
   WHERE matricula_professor = mat;

   SELECT count(*) INTO qtd_alun
   FROM aluno WHERE matricula_aluno = mat;

   retorno := 0;
   IF qtd_prof > 0 THEN
      SELECT
         COUNT(matricula_professor)*800*6+retorno
         INTO retorno FROM ministra
         WHERE matricula_professor = mat
         AND ano_semestre = semestre;

      SELECT
         COUNT(matricula_professor)*137*6+retorno
         INTO retorno FROM disciplina
         WHERE matricula_professor = mat;
```

```
   SELECT
      COUNT(DISTINCT matricula_lider)*106+retorno
      INTO retorno FROM professor
      WHERE matricula_lider = mat;

ELSIF qtd_alun > 0 THEN
   SELECT COUNT(matricula_aluno)*53*4+retorno
      INTO retorno FROM monitoria
      WHERE matricula_aluno = mat
         AND ano_semestre = semestre;

   SELECT COUNT(a_t.matricula_aluno)*102+retorno
      INTO retorno
      FROM aluno_turma a_t, projeto p
      WHERE a_t.matricula_aluno = mat
         AND ano_semestre = semestre
         AND a_t.codigo_projeto = p.codigo_projeto
         AND p.conceito = 'BOM';
```

```
      SELECT (
            CASE
               WHEN nota_vestibular >= 8 THEN 27
               WHEN nota_vestibular >= 5 THEN 15
               ELSE 0
            END) * 6 + retorno INTO retorno
      FROM aluno
      WHERE matricula_aluno = mat;
   END IF;

   RETURN retorno;
END;
/


--TESTANDO
--professor
SELECT calcula_salario(1111,'2010.2') FROM DUAL;
--aluno
SELECT calcula_salario(8888,'2010.2') FROM DUAL;
```

Exercício 13

- **Suponha que existe um imposto a ser cobrado retroativamente dos professores. Numa CONSULTA, utilize a função implementada anteriormente e imprima matrícula, ano_semestre, o valor recebido por cada professor em todos os períodos e o valor do imposto cobrado. Ordene as respostas por período e matrícula.**
- **Os impostos seguirão as seguintes regras: caso o valor do salário do professor seja até R$ 5.000,00 ele pagará 2% de imposto; acima disto até R$ 10.000,00 ele pagará um imposto de 5%; a partir de R$ 10.000,00 o imposto é de 7%. (Use PL diretamente no SELECT – SIM, É POSSÍVEL!)**

```
SELECT pr.matricula_professor,
       tab.ano_semestre,
       calcula_salario(pr.matricula_professor,
                               tab.ano_semestre) AS salario,
       (CASE
          WHEN calcula_salario(pr.matricula_professor,
                               tab.ano_semestre) <= 5000
             THEN calcula_salario(pr.matricula_professor,
                                  tab.ano_semestre)*0.02
          WHEN calcula_salario(pr.matricula_professor,
                               tab.ano_semestre) <= 10000
             THEN calcula_salario(pr.matricula_professor,
                                  tab.ano_semestre)*0.05
          ELSE
                calcula_salario(pr.matricula_professor,
                                  tab.ano_semestre)*0.07
          END) AS imposto
FROM professor pr
   JOIN pessoa pe
      ON (pr.matricula_professor = pe.matricula_pessoa),
   (SELECT DISTINCT ano_semestre FROM TURMA) tab
ORDER BY tab.ano_semestre, pr.matricula_professor;
```

Exercício 14

- **Considerando o modelo de relatório do SIGA, implemente um procedimento que recebe como entrada um número de matrícula de um aluno, um código de uma disciplina, um código de curso e um semestre.**
  O procedimento deve exibir todas as notas (inclusive a final, se houver), a média das notas (não incluindo a final), e a média final (caso necessário). Se os dados de entrada não encontrarem nenhum registro de matrícula ou se o aluno não tiver a nota final mesmo quando precise, então deverão ser tratadas as exceções (Utilize EXCEPTION WHEN).
- **Caso a média seja igual ou superior a 7,0, o aluno receberá um status de "APROVADO POR MÉDIA". Caso contrário, deve-se realizar a média entre a média e a nota final. Se a nota obtida for maior ou igual a 5,0 o status será "APROVADO", se for inferior será "REPROVADO".**

- **Se o aluno não possuir pelo menos 2 notas (excluindo-se a final) ele deverá receber o status ("REPROVADO POR FALTA").**

- **Se o aluno não possuir pelo menos 2 notas (excluindo-se a final) ele deverá receber o status ("REPROVADO POR FALTA").**

```
CREATE OR REPLACE PROCEDURE calcula_notas
   (mat prova.matricula_aluno%TYPE,
   cod_curso prova.codigo_curso%TYPE,
   cod_disciplina prova.codigo_disciplina%TYPE,
   sem prova.ano_semestre%TYPE) IS

   media NUMBER;  final NUMBER;
   cont NUMBER := 0;

BEGIN
   SELECT matricula_aluno INTO media
   FROM aluno_turma
   WHERE matricula_aluno = mat
      AND codigo_disciplina = cod_disciplina
      AND codigo_curso = cod_curso
      AND ano_semestre = sem;
```

```
FOR registro_prova IN
   (SELECT nota, descricao FROM prova
   WHERE matricula_aluno = mat
      AND codigo_disciplina = cod_disciplina
      AND codigo_curso = cod_curso
      AND ano_semestre = sem
      AND descricao <> 'FINAL'
   ORDER BY descricao ASC) LOOP

      cont := cont +1;
      dbms_output.put_line(registro_prova.descricao
         || ': ' ||registro_prova.nota);

   END LOOP;

   IF (cont < 2) THEN
      dbms_output.put_line('STATUS:
                              REPROVADO POR FALTA');
```

```
ELSE
   SELECT AVG(nota) INTO media FROM prova
   WHERE matricula_aluno = mat
      AND codigo_disciplina = cod_disciplina
      AND codigo_curso = cod_curso
      AND ano_semestre = sem
      AND descricao <> 'FINAL';

   dbms_output.put_line('MEDIA: ' ||media);

      IF (media >= 7) THEN
         dbms_output.put_line('STATUS:
                                  APROVADO POR MEDIA');
      ELSE
         SELECT nota INTO final FROM prova
         WHERE matricula_aluno = mat
            AND codigo_disciplina = cod_disciplina
            AND codigo_curso = cod_curso
            AND ano_semestre = sem
            AND descricao = 'FINAL';
```

```
dbms_output.put_line('FINAL: '||final);
dbms_output.put_line('MEDIA FINAL: '||
                              ((media+final)/2));

IF ((media+final)/2 < 5) THEN
   dbms_output.put_line('STATUS:REPROVADO');
      ELSE
         dbms_output.put_line('STATUS: APROVADO');
      END IF;
   END IF;
END IF;
```

```
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         IF (media IS NULL) THEN
            RAISE_APPLICATION_ERROR(-20101,'NAO HA
                        REGISTROS DE MATRICULA
                        PARA O ALUNO E DISCIPLINA
                        ESPECIFICADOS');
         ELSIF (final IS NULL) THEN
            RAISE_APPLICATION_ERROR(-20101,'A NOTA
                        DA PROVA FINAL AINDA ESTA
                        PENDENTE');
         END IF;
END;
/


--TESTANDO
EXECUTE calcula_notas(2626,3,2,'2010.2');
```

Exercício 15

- **Adapte o procedimento anterior e implemente um TRIGGER de linha que seja disparado quando se quiser cadastrar um aluno numa monitoria. É necessário observar-se a tentativa de pagar a cadeira mais recente do aluno .**
- **Neste caso, não é necessário imprimir nenhum nota, mas sim levantar-se uma exceção que indique, caso haja, a impossibilidade do cadastro e o status que motivou isso.**
- **Admita que um aluno precisa de pelo menos 2 provas (desconsiderando-se a final) para ter sua situação definida. Caso o aluno não tenha ainda 2 provas ou tenha obtido média abaixo de 7,0 , mas ainda não tiver realizado a final indique o status para "INDEFINIDO"**

```
CREATE OR REPLACE TRIGGER controle_calcula_notas
AFTER INSERT ON monitoria
FOR EACH ROW

DECLARE
   media NUMBER;
   final NUMBER;
   cont NUMBER := 0;

BEGIN
   SELECT COUNT(nota), AVG(nota) INTO cont, media
   FROM prova
   WHERE matricula_aluno = :NEW.matricula_aluno
   AND codigo_disciplina = :NEW.codigo_disciplina
   AND codigo_curso = :NEW.codigo_curso
   AND descricao <> 'FINAL'
   GROUP BY ano_semestre
   HAVING ano_semestre = MAX(ano_semestre);
```

```
IF cont < 2 THEN
   RAISE_APPLICATION_ERROR(-20102,'A SITUACAO DO
                    ALUNO AINDA ESTA INDEFINIDA');
ELSE
   IF (media < 7) THEN
      SELECT SUM(nota) INTO final FROM prova
      WHERE matricula_aluno = :NEW.matricula_aluno
        AND codigo_disciplina =
                              :NEW.codigo_disciplina
        AND codigo_curso = :NEW.codigo_curso
        AND descricao = 'FINAL'
      GROUP BY ano_semestre
      HAVING ano_semestre = MAX(ano_semestre);

        IF ((media+final)/2 < 5) THEN
        RAISE_APPLICATION_ERROR(-20102,'O ALUNO
                FOI REPROVADO POR MEDIA');
      END IF;
   END IF;
END IF;
```

```
EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR(-20103,'A
                  SITUACAO DO ALUNO AINDA ESTA
                  INDEFINIDA');
END;
/


--TESTANDO
EXECUTE calcula_notas(3030,3,2,'2010.2');

INSERT INTO monitoria (codigo_disciplina,
   codigo_curso, ano_semestre, matricula_aluno,
   matricula_professor)
   VALUES  (2,3,'2011.1',3030,1111);
```

Exercício 16

- **Crie uma tabela chamada log_provas com os campos 'tipo_de_acao VARCHAR2' e 'hora TIMESTAMP'.**
- **Implemente um TRIGGER para coletar um log na tabela log_provas todas as vezes que alguma ação ocorrer na tabela provas.**
- **Após a operação verifique qual foi o tipo da ação e insira um novo registro na tabela (utilize SYSDATE para preencher a hora). O que importa em si é apenas a operação, e não cada procedimento que ela executa (escolha o tipo de TRIGGER adequadamente).**

Exercício 16

- Crie uma tabela chamada log_provas com os campos 'tipo_de_acao VARCHAR2' e 'hora TIMESTAMP'.
- Implemente um TRIGGER para coletar um log na tabela log_provas todas as vezes que alguma ação ocorrer na tabela provas.
- Após a operação verifique qual foi o tipo da ação e insira um novo registro na tabela (utilize SYSDATE para preencher a hora). O que importa em si é apenas a operação, e não cada procedimento que ela executa (escolha o tipo de TRIGGER adequadamente).

```
CREATE TABLE log_provas(
   tipo_de_acao VARCHAR2(15),
   hora TIMESTAMP
);
```

```
CREATE OR REPLACE TRIGGER controle_de_log
        AFTER INSERT OR UPDATE OR DELETE ON prova
BEGIN
   IF(INSERTING) THEN
      INSERT INTO log_provas(tipo_de_acao, hora)
                  VALUES ('INSERCAO', SYSDATE);

   ELSIF (UPDATING) THEN
      INSERT INTO log_provas(tipo_de_acao, hora)
                  VALUES ('ATUALIZACAO', SYSDATE);

   ELSIF (DELETING) THEN
      INSERT INTO log_provas(tipo_de_acao, hora)
                  VALUES ('REMOCAO', SYSDATE);
   END IF;
END;
/
```

Exercício 17

- **Implemete um TRIGGER que regulamentará as matriculas.**
- **Admita que para uma cadeira NÃO ser considerada eletiva, ela precisa ter sido ofertada pelo menos 3 vezes consecutivas em quaisquer períodos.**
- **Use a função que calcula a quantidade de créditos e não permita que alunos que possuem menos de 15 créditos possam se matricular em cadeiras eletivas.**
- **Ainda, para vincular-se a quaisquer projetos, estes não podem ter sido anteriormente utilizados em outras turmas nem podem estar vinculados a mais do que 4 alunos.**

```
CREATE OR REPLACE TRIGGER controle_matricula
   BEFORE INSERT ON aluno_turma
FOR EACH ROW
DECLARE

   contador NUMBER; creditos NUMBER;
   ano1 NUMBER; semestre1 NUMBER; ano2 NUMBER;
   semestre2 NUMBER; ano3 NUMBER; semestre3 NUMBER;

   CURSOR cursor_periodos IS
   SELECT t.ano_semestre FROM turma t
   WHERE t.codigo_disciplina =
                              :NEW.codigo_disciplina
   AND t.codigo_curso = (
      SELECT a.codigo_curso FROM aluno a
      WHERE a.matricula_aluno = :NEW.matricula_aluno
   ) ORDER BY t.ano_semestre DESC;

   periodo turma.ano_semestre%TYPE;
```

```
BEGIN

creditos := qtd_creditos(:NEW.matricula_aluno);
   IF creditos < 15 THEN

      OPEN cursor_periodos;
      FETCH cursor_periodos INTO periodo;
      desmembra_semestre(periodo, ano1, semestre1);

      FETCH cursor_periodos INTO periodo;
      desmembra_semestre(periodo, ano2, semestre2);

      FETCH cursor_periodos INTO periodo;
      desmembra_semestre(periodo, ano3, semestre3);
```

```
IF(cursor_periodos%ROWCOUNT = 3) THEN
     IF NOT(
(ano1 = ano2 AND ano3 = ano1-1 AND semestre3 = 2)
                          OR
(ano2 = ano3 AND ano1 = ano2+1 AND semestre1 = 2))
     THEN --consecutivos

        RAISE_APPLICATION_ERROR(-20105,'ALUNO NAO
            PODE PAGAR CADEIRA ELETIVA.  NAO
            CONSECUTIVA');
     END IF;

   ELSE
     RAISE_APPLICATION_ERROR(-20105,'ALUNO NAO
          PODE PAGAR CADEIRA ELETIVA. MENOS DE 3
          OFERTAS');--nao foi paga nem tres vezes
   END IF;

   CLOSE cursor_periodos;
END IF;
```

```
   SELECT COUNT(codigo_projeto) INTO contador
   FROM aluno_turma
   WHERE codigo_projeto = :NEW.codigo_projeto
   GROUP BY ano_semestre, codigo_disciplina,
      codigo_curso;

   IF(contador >= 4) THEN
      RAISE_APPLICATION_ERROR(-20105,'ESTE PROJETO
                  JA ESTA COM O NUMERO DE ALUNOS
COMPLETO');
   END IF;

   EXCEPTION
      WHEN TOO_MANY_ROWS THEN
         RAISE_APPLICATION_ERROR(-20105,'ESTE
                     PROJETO JA FOI UTILIZADO EM
                     OUTRA TURMA');
END;
/
```