Monitoria GDI
Aula Prática

DML + PL/SQL

parte 1

DML

linguagem de manipulação de dados

SQL

- **Estrutura básica de uma consulta SQL**

```
SELECT Coluna1[,Coluna2[,Coluna3[,...]]]
FROM Tabela1[,Tabela2[,...]]
WHERE Condição
```

SQL

- **Estrutura genérica de uma consulta SQL**

```
SELECT [DISTINCT|ALL] {*|[Tabela.]Coluna1 [AS Alias1]
                [[Tabela.]Coluna2 [AS Alias2] [,...]]}
FROM Tabela1[,Tabela2[,...]]
[WHERE {Condição Simples|Condição de Sub-consulta}]
[ORDER BY Coluna1 [ASC|DESC] [,Coluna2 [ASC|DESC] [, ... ]]]
[GROUP BY Coluna1 [,Coluna2[,...]] [HAVING Condição]]
[{UNION|INTERSECT|EXCEPT} SELECT ...]
```

Exercício 1

- **Selecione a matrícula e o nome de todas as mulheres, ordenando-as por ordem alfabética.**

Exercício 1

- **Selecione a matrícula e o nome de todas as mulheres, ordenando-as por ordem alfabética.**

```
SELECT nome, matricula_pessoa
FROM Pessoa
WHERE Sexo = 'M'
ORDER BY nome;
```

Exercício 2

- **Repita a operação anterior exibindo apenas aquelas que são professoras.**

Exercício 2

- **Repita a operação anterior exibindo apenas aquelas que são professoras.**

```
SELECT nome, matricula_pessoa
FROM pessoa, professor
WHERE matricula_pessoa = matricula_professor
   AND Sexo = 'M'
ORDER BY nome;
```

Exercício 3

- **Quais são as disciplinas que o professor 'Sirenio Arruda' está ministrando atualmente?**

Exercício 3

- **Quais são as disciplinas que o professor 'Sirenio Arruda' está ministrando atualmente?**

```
SELECT M.codigo_disciplina
FROM Ministra M, Pessoa P, Professor PR
WHERE M.matricula_professor = PR.matricula_professor
   AND PR.matricula_professor = P.matricula_pessoa
   AND P.nome = 'Sirenio Arruda'
   AND M.ano_semestre = '2010.2';
```

Exercício 4

- **Repita a consulta anterior utilizando JOIN.**

Exercício 4

- **Repita a consulta anterior utilizando JOIN.**

```
SELECT M.codigo_disciplina
FROM Ministra M
   INNER JOIN Professor PR
      ON M.matricula_professor = PR.matricula_professor
   INNER JOIN Pessoa P
      ON PR.matricula_professor = P.matricula_pessoa
WHERE P.nome = 'Sirenio Arruda'
   AND M.ano_semestre = '2010.2';
```

Exercício 5

- **Para as disciplinas de código 1, 2 e 3, mostre quais alunos já foram seus monitores. (Use IN)**

Exercício 5

- **Para as disciplinas de código 1, 2 e 3, mostre quais alunos já foram seus monitores. (Use IN)**

```
SELECT DISTINCT P.nome
FROM Pessoa P
   INNER JOIN Aluno A
      ON P.matricula_pessoa = A.matricula_aluno
   INNER JOIN Monitoria M
      ON M.matricula_aluno = A.matricula_aluno
   WHERE M.codigo_disciplina IN (1,2,3);
```

Exercício 6

- **Mostre os nomes de TODOS os professores e, caso existam, os nomes dos seus líderes.**

Exercício 6

- **Mostre os nomes de TODOS os professores e, caso existam, os nomes dos seus líderes.**

```
SELECT P1.nome, P2.nome as lider
FROM Pessoa P1
   INNER JOIN Professor PR
      ON P1.matricula_pessoa = PR.matricula_professor
   LEFT OUTER JOIN Pessoa P2
      ON PR.matricula_lider = P2.matricula_pessoa;
```

Exercício 7

- **Mostre os alunos que não têm nenhum projeto. Exiba também as informações de quando eles pagaram a cadeira. (Use IS NULL)**

Exercício 7

- **Mostre os alunos que não têm nenhum projeto. Exiba também as informações de quando eles pagaram a cadeira. (Use IS NULL)**

```
SELECT a.matricula_aluno,
   at.codigo_curso,
   at.codigo_disciplina,
   at.ano_semestre
FROM aluno a, aluno_turma at
WHERE a.matricula_aluno = at.matricula_aluno
   AND at.codigo_projeto IS NULL
ORDER BY a.matricula_aluno, at.ano_semestre;
```

Exercício 8

- **Selecione todos os professores, exceto aqueles que entraram entre 1995 e 2005. (Use BETWEEN)**

Exercício 8

- **Selecione todos os professores, exceto aqueles que entraram entre 1995 e 2005. (Use BETWEEN)**

```
SELECT *
FROM professor
WHERE data_admissao NOT BETWEEN
   to_date('1999', 'yyyy') AND to_date('2005', 'yyyy');
```

Exercício 9

- **Mostre quantas vezes que o professor 'Jose Alcantara' já esteve a lecionar**

Exercício 9

- **Mostre quantas vezes que o professor 'Jose Alcantara' já esteve a lecionar**

```
SELECT COUNT(M.codigo_disciplina)
FROM Pessoa P
    INNER JOIN Professor PR
       ON P.matricula_pessoa = PR.matricula_professor
    INNER JOIN Ministra M
       ON M.matricula_professor = PR.matricula_professor
WHERE P.nome = 'Jose Alcantara';
```

Exercício 10

- **Mostre a média das notas dos alunos agrupadas por período.**

Exercício 10

- **Mostre a média das notas dos alunos agrupadas por período.**

```
SELECT ano_semestre, AVG(nota)
FROM Prova
GROUP BY ano_semestre;
```

Exercício 11

- **Considere um relatório e mostre, numa mesma consulta, para o semeste '2009.1', os registros dos professores em todas as ministrações que realizaram mais os registros dos alunos nas vezes em que pagaram alguma cadeira. Exiba o código da disciplina, o código do curso e a matrícula do professor ou do aluno que realizou a atividade. (Realize SELECTS independentes e use UNION)**

Exercício 11

- **Considere um relatório e mostre, numa mesma consulta, para o semeste '2009.1', os registros dos professores em todas as ministrações que realizaram mais os registros dos alunos nas vezes em que pagaram alguma cadeira. Exiba o código da disciplina, o código do curso e a matrícula do professor ou do aluno que realizou a atividade. (Realize SELECTS independentes e use UNION)**

```
(SELECT matricula_professor AS matricula,
   codigo_disciplina, codigo_curso  FROM ministra
   WHERE ano_semestre = '2009.1')
                  UNION
   (SELECT matricula_aluno AS matricula,
   codigo_disciplina, codigo_curso FROM aluno_turma
   WHERE ano_semestre = '2009.1');
```

PL/SQL

Procedural Language/ Structured Query Language

PROCEDURE

- **Por padrão não retornam valor (exceção: modo OUT ou IN OUT).**

- **Estrutura básica de um PROCEDURE**

```
PROCEDURE nome IS
BEGIN
   [EXCEPTION]
END;
```

FUNCTION

- **Por padrão, necessariamente, retornam um único valor.**
- **Estrutura básica de uma FUNCTION**

```
FUNCTION nome RETURN tipo IS
BEGIN
   RETURN valor
   [EXCEPTION]
END;
```

Exercício 12

- **Admita que cada uma das cadeiras que um aluno paga vale 5 créditos, que cada projeto vale 1 e que cada monitoria vale 2 créditos.**
- **Implemente uma função que, dado um número de matrícula, retorna os créditos totais da carreira estudantil do aluno.**

Exercício 12

- **Admita que cada uma das cadeiras que um aluno paga vale 5 créditos, que cada projeto vale 1 e que cada monitoria vale 2 créditos.**
- **Implemente uma função que, dado um número de matrícula, retorna os créditos totais da carreira estudantil do aluno.**

```
CREATE OR REPLACE FUNCTION qtd_creditos
   (mat aluno.matricula_aluno%TYPE)
   RETURN NUMBER IS

   retorno NUMBER;
```

```
BEGIN
   SELECT COUNT(a_t.matricula_aluno)*5
      + COUNT(a_t.codigo_projeto)*1
      + COUNT(m.matricula_aluno)*2 INTO retorno
   FROM aluno_turma a_t, monitoria m, aluno a
   WHERE a.matricula_aluno = a_t.matricula_aluno
      AND m.matricula_aluno = a.matricula_aluno
      AND a.matricula_aluno = mat;

   RETURN retorno;
END;
/


--TESTANDO
SELECT qtd_creditos(9999) FROM DUAL;
```

Exercício 13

- **Implemente um procedimento que recebe como parâmetro de entrada um título de um projeto e imprime os seus dados.**

Exercício 13

- **Implemente um procedimento que recebe como parâmetro de entrada um título de um projeto e imprime os seus dados.**

```
CREATE OR REPLACE PROCEDURE pesquisa_projeto
(par_titulo IN projeto.titulo%TYPE) IS

   v_codigo_projeto projeto.codigo_projeto%TYPE;
   v_titulo projeto.titulo%TYPE;
   v_conceito projeto.conceito%TYPE;
   v_hp projeto.hp%TYPE;
```

```
BEGIN
   SELECT codigo_projeto, titulo, conceito, hp
      INTO v_codigo_projeto,v_titulo,v_conceito,v_hp
      FROM projeto
      WHERE titulo LIKE par_titulo;

   dbms_output.put_line(
      'COD: ' ||  v_codigo_projeto||
      ' - TIT: ' || v_titulo ||
      ' - CON: ' || v_conceito ||
      ' - HP: ' || v_hp);
END;
/


--TESTANDO
EXECUTE pesquisa_projeto('Rede Aberta');
```

Exercício 14

- **Implemente um novo procedimento, semelhante ao anterior, que seja mais genérico e pesquise todos os projetos que possuam o valor do parâmetro como substring do seu título. (Utilize LIKE '%' e CURSOR)**

Exercício 14

- **Implemente um novo procedimento, semelhante ao anterior, que seja mais genérico e pesquise todos os projetos que possuam o valor do parâmetro como substring do seu título. (Utilize LIKE '%' e CURSOR)**

```
CREATE OR REPLACE PROCEDURE
                        pesquisa_projeto_generico
   (par_titulo IN projeto.titulo%TYPE) IS

   CURSOR cursor_projetos IS
      SELECT *
      FROM projeto
      WHERE LOWER(titulo)
         LIKE LOWER('%'||par_titulo||'%');

   registro_projeto projeto%ROWTYPE;
```

```
BEGIN
   OPEN cursor_projetos;
   LOOP
      FETCH cursor_projetos INTO registro_projeto;
      EXIT WHEN cursor_projetos%NOTFOUND;
         dbms_output.put_line(
      'COD: '||registro_projeto.codigo_projeto||
      ' - TIT: ' || registro_projeto.titulo||
      ' - CON: ' || registro_projeto.conceito||
      ' - HP: ' || registro_projeto.hp);
   END LOOP;
   CLOSE cursor_projetos;
END;
/


--TESTANDO
EXECUTE pesquisa_projeto_generico('cin');
```

Exercício 15

- **Crie um PROCEDURE que recebe um VARCHAR do tipo ano_semestre e produz dois parâmetros numéricos de saída: ano e semestre;**

Exercício 15

- **Crie um PROCEDURE que recebe um VARCHAR do tipo ano_semestre e produz dois parâmetros numéricos de saída: ano e semestre;**

```
CREATE OR REPLACE PROCEDURE desmembra_semestre
   (ano_semestre IN turma.ano_semestre%TYPE,
   ano OUT NUMBER,
   semestre OUT NUMBER) IS

BEGIN
   ano := SUBSTR(ano_semestre,1,4);
   semestre := SUBSTR(ano_semestre,6,1);
END;
/
```

Exercício 16

- **Implemente uma FUNCTION que receberá o código de uma disciplina e retornará uma STRING com todos os ANOS em que ela foi ofertada no 1º semestre e todos os anos para o 2º semestre (EX: '1º: 1992; 1990; 2000; 2º: 1990; 2001;').**
- **Crie uma tabela (IS TABLE OF) com registros do tipo (IS RECORD [cod_curso, ano, semestre]) que receberá as informações de todas as turmas que já existiram e utilize o PROCEDURE anterior para separar os campos ano_semestre.**
- **Em seguida, verifique um a um os registros da tabela já povoada e vá preenchendo a variável de retorno.**

```
CREATE OR REPLACE FUNCTION anos_por_semestre
   (cod disciplina.codigo_disciplina%TYPE)
   RETURN VARCHAR2 IS

   TYPE TIPO_TURMA_COMPACTO IS RECORD
      (cod_curso turma.codigo_curso%TYPE,
      ano NUMBER, semestre NUMBER);

   TYPE TIPO_TURMA_COMPACTO_TABELA
      IS TABLE OF TIPO_TURMA_COMPACTO;

   tab_compacto TIPO_TURMA_COMPACTO_TABELA :=
                        TIPO_TURMA_COMPACTO_TABELA();

   anos1 VARCHAR2(100) := '1°: ';
   anos2 VARCHAR2(100) := '2°: ';
```

```
BEGIN
   FOR registro_turma IN
        (SELECT * FROM turma
        WHERE codigo_disciplina = cod
        ORDER BY codigo_curso) LOOP

      tab_compacto.EXTEND;
      tab_compacto(tab_compacto.LAST).cod_curso :=
                        registro_turma.codigo_curso;

   desmembra_semestre(registro_turma.ano_semestre,
        tab_compacto(tab_compacto.LAST).ano,
        tab_compacto(tab_compacto.LAST).semestre);

   END LOOP;
```

```
WHILE tab_compacto.COUNT > 0 LOOP
   IF(tab_compacto(tab_compacto.LAST).semestre = 1) THEN
      anos1 := anos1 ||
      tab_compacto(tab_compacto.LAST).ano || '-'||
      tab_compacto(tab_compacto.LAST).cod_curso ||'; ';
   ELSIF(tab_compacto(tab_compacto.LAST).semestre = 2) THEN
      anos2 := anos2
      || tab_compacto(tab_compacto.LAST).ano || '-'||
      tab_compacto(tab_compacto.LAST).cod_curso ||'; ';
   END IF;

   tab_compacto.TRIM();

END LOOP;

   RETURN anos1 ||'  '||anos2;

END;
/

--TESTANDO
SELECT anos_por_semestre(1) FROM DUAL;
```

TRIGGER

- **Executado implicitamente pelo SGBD na ocorrência de um determinado evento ou combinação deste.**

- **Estrutura básica de um TRIGGER**

```
CREATE [OR REPLACE] TRIGGER nome_trigger
      momento evento1 [OR evento2 OR evento3]
       [OF coluna] ON nome_objeto
[[REFERENCING OLD AS apelido1 | NEW AS apelido2]

FOR EACH ROW

[WHEN (condição)]]

corpo_trigger
```

Exercício 17

- **Criar um TRIGGER que faça um comparativo entre os ANTIGOS e NOVOS valores logo após inserção, atualização ou deleção de um projeto.**

Exercício 17

- **Criar um TRIGGER que faça um comparativo entre os ANTIGOS e NOVOS valores logo após inserção, atualização ou deleção de um projeto.**

```
CREATE OR REPLACE TRIGGER controle_projetos
   AFTER INSERT OR UPDATE OR DELETE ON PROJETO
FOR EACH ROW

BEGIN
   dbms_output.put_line('<<Dados ANTIGOS>>');
   dbms_output.put_line('COD: '||
                              :OLD.codigo_projeto);
   dbms_output.put_line('TIT: '||:OLD.titulo);
   dbms_output.put_line('CON: '||:OLD.conceito);
   dbms_output.put_line('HP: '||:OLD.hp);
   dbms_output.put_line(' ');
```

```
   dbms_output.put_line('<<Dados NOVOS>>');
   dbms_output.put_line('COD: '||
                                :NEW.codigo_projeto);
   dbms_output.put_line('TIT: '||:NEW.titulo);
   dbms_output.put_line('CON: '||:NEW.conceito);
   dbms_output.put_line('HP: '||:NEW.hp);
END;
/


--TESTANDO
INSERT INTO projeto(codigo_projeto,titulo,
   conceito,hp) VALUES (21,'BiosFera','RUIM',
   'www.cin.ufpe.br/~biosfera');
UPDATE projeto SET titulo = 'Bioma Protection',
   hp = 'www.biomaprotection.com',
   conceito = 'BOM'  WHERE codigo_projeto = 21;
DELETE projeto WHERE codigo_projeto = 21;
```

Exercício 18

- **Implemente um TRIGGER que não permita que um professor coordene mais do que uma disciplina. Caso alguma irregularidade ocorra, imprima uma mensagem do tipo "RAISE APPLICATION ERROR".**

Exercício 18

- **Implemente um TRIGGER que não permita que um professor coordene mais do que uma disciplina. Caso alguma irregularidade ocorra, imprima uma mensagem do tipo "RAISE APPLICATION ERROR".**

```
CREATE OR REPLACE TRIGGER controle_coordenacao
BEFORE INSERT ON disciplina
FOR EACH ROW

DECLARE
   coordenador disciplina.matricula_professor%TYPE;
```

```
BEGIN
   SELECT matricula_professor INTO coordenador
   FROM disciplina
   WHERE matricula_professor =
                          :NEW.matricula_professor;
   IF coordenador IS NOT NULL THEN
      RAISE_APPLICATION_ERROR(-20101,'ESTE PROFESSOR
                          JA COORDENA UMA DISCIPLINA');
   END IF;

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         dbms_output.put_line('COORDENACAO ACEITA');
END;
/

--TESTANDO
INSERT INTO disciplina (codigo_disciplina,
   ementa, conteudo_programatico,
   matricula_professor)VALUES (7,'E7', 'C7',1111);
```

Monitoria GDI
Aula Prática

DML + PL/SQL

parte 1