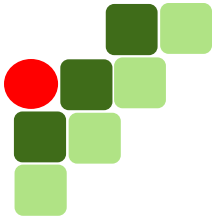
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Araraquara

Banco de Dados II

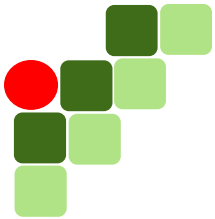
Cristiane Yaguinuma
cristiane.yaguinuma@ifsp.edu.br

- Comandos PL/SQL



Roteiro da aula

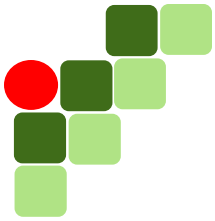
- PL/SQL
  - Registros
  - Processamento condicional
  - Processamento repetitivo



Registros

- Tipos de dados compostos por campos
  - cada campo possui um tipo de dado

- Usados para obter dados de **uma tupla** para processamento em PL/SQL

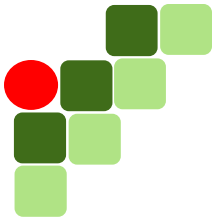


Registros

```
TYPE tipo_registro IS RECORD
      (campo1[, campo2]…);
```

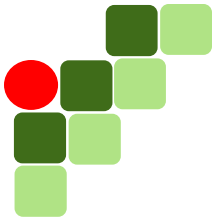
- Campos podem ser declarados como
  - Tipo de dados
  - variavel%TYPE
  - tabela.coluna%TYPE
  - tabela%ROWTYPE

```
DECLARE
      nome_variavel tipo_registro;
```



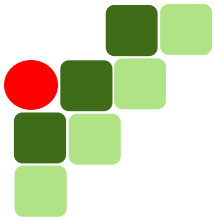
Registros

```
DECLARE
        TYPE emp_registro IS RECORD
                (last_name VARCHAR2(25),
                 job_id VARCHAR2(10),
                 salary NUMBER(8,2));
        dados_emp emp_registro;
BEGIN
        SELECT last_name, job_id, salary INTO dados_emp
        FROM employees
        WHERE employee_id = 100;

        DBMS_OUTPUT.PUT_LINE ('Empregado: '|| dados_emp.last_name || '
        Função: ' || dados_emp.job_id || '
        Salário: ' || dados_emp.salary);
END;
/
```



Atributo %ROWTYPE

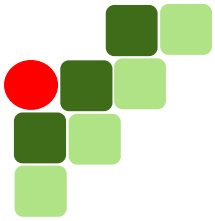
```
DECLARE
   emp_rec employees%ROWTYPE;
   emp_id employees.employee_id%TYPE := 100;
BEGIN
   SELECT * INTO emp_rec FROM employees
   WHERE employee_id = emp_id;

   DBMS_OUTPUT.PUT_LINE ('Empregado: '||
emp_rec.employee_id || '
Nome: ' || emp_rec.last_name || '
Data de contratação: ' || emp_rec.hire_date || '
Salário: ' || emp_rec.salary || '
Departamento: ' || emp_rec.department_id);
END;
/
```



Unidade de programa PL/SQL

- Um bloco possui a seguinte estrutura:

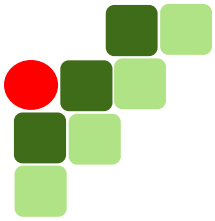
```
[DECLARE]
   -- declaração de variáveis, constantes
   -- contém inicializações
BEGIN
   -- comandos SQL
   -- estruturas de programação em PL/SQL
[EXCEPTION]
   -- tratamento de erros e emissão de mensagens
END;
```



Comandos PL/SQL

- Processamento Condicional:
  - IF
  - CASE

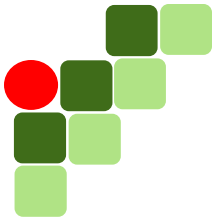
- Processamento Repetitivo
  - LOOP
  - WHILE
  - FOR



Comando IF-THEN-ELSE

```
IF condição1 THEN
      Comandos executados caso a condição1 seja verdadeira
   [ELSIF condição2 THEN
      Comandos executados caso a condição2 seja verdadeira
   ]
   [ELSE
      Comandos executados caso nenhuma condição seja
verdadeira
   ]
END IF;
```

```
DECLARE
  nota CHAR(1) := UPPER('&nota');
  resultado VARCHAR2(20);
BEGIN
   IF nota = 'A' THEN
      resultado := 'Excelente';
   ELSIF nota = 'B' THEN
      resultado := 'Muito bom';
   ELSIF nota = 'C' THEN
       resultado := 'Bom';
   ELSIF nota IN ('D', 'E') THEN
       resultado := 'Reprovado';
   ELSE
       resultado := 'Nota inválida';
   END IF;
  DBMS_OUTPUT.PUT_LINE ('Nota: '|| nota || '
Resultado: ' || resultado);
END;
/
```

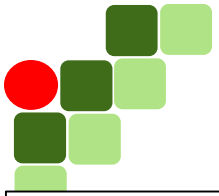


Comando CASE

```
CASE seletor
   WHEN expressao1 THEN resultado1
   WHEN expressao2 THEN resultado2
   ...
   WHEN expressaoN THEN resultadoN
   [ELSE resultadoN+1]
END CASE;
```

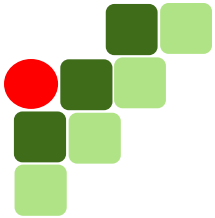
Comando CASE

```
DECLARE
  nota CHAR(1) := UPPER('&nota');
  resultado VARCHAR2(20);
BEGIN
  resultado :=
    CASE nota
      WHEN 'A' THEN 'Excelente'
      WHEN 'B' THEN 'Muito bom'
      WHEN 'C' THEN 'Bom'
      ELSE 'Nota inválida'
    END CASE;
  DBMS_OUTPUT.PUT_LINE ('Nota: '|| nota || '
Resultado: ' || resultado);
END;
/
```



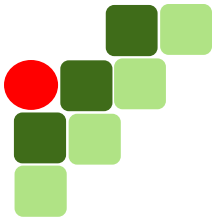
Comando CASE

```
DECLARE
  nota CHAR(1) := UPPER('&nota');
  resultado VARCHAR2(20);
BEGIN
  resultado :=
    CASE
      WHEN nota = 'A' THEN 'Excelente'
      WHEN nota = 'B' THEN 'Muito bom'
      WHEN nota = 'C' THEN 'Bom'
      WHEN nota IN ('D', 'E') THEN 'Reprovado'
      ELSE 'Nota inválida'
    END CASE;
  DBMS_OUTPUT.PUT_LINE ('Nota: '|| nota || '
Resultado: ' || resultado);
END;
/
```



Processamento repetitivo

- LOOP
- WHILE
- FOR

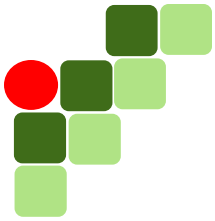


Comando LOOP

- LOOP: repete a execução de comandos até que seja atingida uma condição

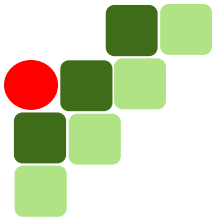
```
LOOP
    comandos que devem ser repetidos
    EXIT WHEN condição;
END LOOP;
```

- Obs.: as variáveis que controlam a iteração devem ser declaradas e explicitamente incrementadas



Comando LOOP

```
DECLARE
  ctry_id locations.country_id%TYPE := 'BR';
  loc_id locations.location_id%TYPE;
  counter NUMBER(2) := 1;
  cty locations.city%TYPE := 'Araraquara';
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = ctry_id;
  LOOP
    INSERT INTO locations(location_id, city, country_id)
      VALUES((loc_id + counter), cty, ctry_id);
    counter := counter + 1;
    EXIT WHEN counter > 3;
  END LOOP;
END;
/
```

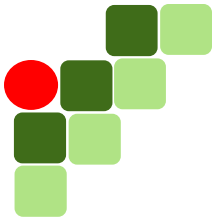


Comando WHILE

- WHILE: efetua a iteração mediante a verificação de uma condição

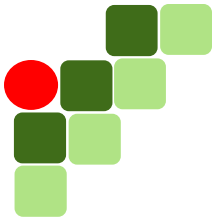
```
WHILE condição LOOP
    comandos que devem ser repetidos
END LOOP;
```

- Obs.: as variáveis que controlam a iteração devem ser declaradas e explicitamente incrementadas



Comando WHILE

```
DECLARE
  ctry_id locations.country_id%TYPE := 'BR';
  loc_id locations.location_id%TYPE;
  counter NUMBER(2) := 1;
  cty locations.city%TYPE := 'Araraquara';
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = ctry_id;
  WHILE counter <= 3 LOOP
    INSERT INTO locations(location_id, city, country_id)
      VALUES((loc_id + counter), cty, ctry_id);
    counter := counter + 1;
  END LOOP;
END;
/
```



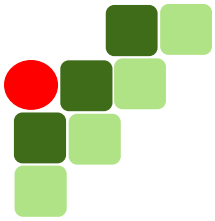
Comando FOR

- FOR: repete n vezes com n conhecido

```
FOR i in 1..n LOOP
   comandos que devem ser repetidos
END LOOP;
```

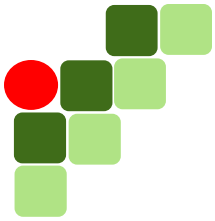
```
FOR i in REVERSE n..1 LOOP
   comandos que devem ser repetidos
END LOOP;
```

  - O contador deve ser referenciado somente dentro do loop
  - Não se deve modificar o valor do contador dentro do loop



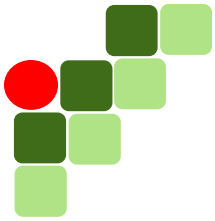
Comando FOR

```
DECLARE
  ctry_id locations.country_id%TYPE := 'BR';
  loc_id locations.location_id%TYPE;
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = ctry_id;
  FOR i IN 0..5 LOOP
    DELETE FROM locations
    WHERE location_id = (loc_id - i);
  END LOOP;
END;
/
```



Recomendações

- Utilize o comando LOOP quando os comandos dentro do loop devem ser executados pelo menos uma vez
- Utilize o comando WHILE se a condição tem que ser avaliada no início de cada iteração
- Utilize o comando FOR se o número de iterações (n) é previamente conhecido



Exercícios de fixação

- Criar um bloco anônimo PL/SQL para reajustar o salário de um determinado empregado conforme sua função (job_id):
  - ‘PU_CLERK’: 12% de reajuste
  - 'SH_CLERK': 11% de reajuste
  - 'ST_CLERK': 10% de reajuste
  - Outros: 5% de reajuste

```
DECLARE
  empdados employees%ROWTYPE;
  empid employees.employee_id%TYPE := 115;
  reajuste NUMBER(3,2);
BEGIN
  SELECT * INTO empdados from employees
  WHERE employee_id = empid;
  IF empdados.job_id = 'PU_CLERK' THEN
       reajuste := .12;
  ELSIF empdados.job_id = 'SH_CLERK' THEN
       reajuste := .11;
  ELSIF empdados.job_id = 'ST_CLERK' THEN
       reajuste := .10;
  ELSE
       reajuste := .5;
  END IF;
  UPDATE employees
  SET salary = empdados.salary + empdados.salary * reajuste
  WHERE employee_id = empid;
END;
/
```

```
DECLARE
  empdados employees%ROWTYPE;
  empid employees.employee_id%TYPE := 115;
  reajuste NUMBER(3,2);
BEGIN
  SELECT * INTO empdados from employees
  WHERE employee_id = empid;
  CASE
       WHEN empdados.job_id = 'PU_CLERK' THEN
              reajuste := .12;
       WHEN empdados.job_id = 'SH_CLERK' THEN
              reajuste := .11;
       WHEN empdados.job_id = 'ST_CLERK' THEN
              reajuste := .10;
       ELSE
              reajuste := .5;
  END CASE;
  UPDATE employees
  SET salary = empdados.salary + empdados.salary * reajuste
  WHERE employee_id = empid;
END;
/
```