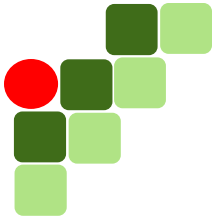
Banco de Dados II

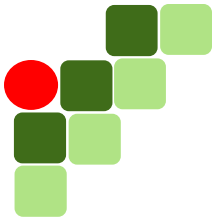
Cristiane Yaguinuma
cristiane.yaguinuma@ifsp.edu.br

- Cursores PL/SQL



Roteiro da aula

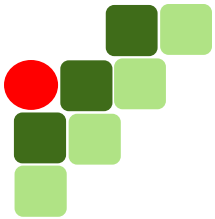
- Cursores Implícitos
- Cursores Explícitos
- Cursores e registros
- Atributos de cursores explícitos
- Exercícios



Cursores

- Um cursor é um ponteiro para uma área de memória privada alocada no SGBD

- Dois tipos de cursores
  - Cursores implícitos
  - Cursores explícitos

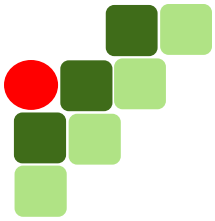


Cursores implícitos

- Declarados e gerenciados internamente pelo SGBD para todo comando DML e SELECT INTO

- Atributos de cursores implícitos – válidos para o comando mais recentemente executado
  - SQL%FOUND
  - SQL%NOTFOUND
  - SQL%ROWCOUNT

```
DECLARE
      mgr_id NUMBER(6) := &mgr_id;
BEGIN
      DELETE FROM employees
      WHERE manager_id = mgr_id;
      IF SQL%FOUND THEN
         DBMS_OUTPUT.PUT_LINE('Quantidade de
empregados removidos: ' || TO_CHAR(SQL%ROWCOUNT));
      ELSE
         DBMS_OUTPUT.PUT_LINE('Não existem
empregados gerenciados pelo empregado com id: ' ||
mgr_id);
     END IF;
END;
/
ROLLBACK;
```

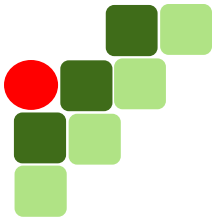


SELECT INTO

- Recuperar dados do BD para variáveis em blocos PL/SQL

```
SELECT coluna1[, coluna2]…
INTO nome_variavel1[, nome_variavel2]…
FROM tabela1[, tabela2]…
[WHERE condições]
```

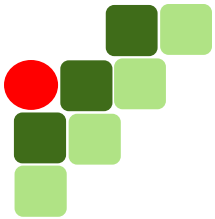
- Consultas devem retornar apenas uma tupla



Recuperando dados do BD com PL/SQL

- E quando as consultas retornam mais de uma tupla?
  - Escreva um bloco anônimo PL/SQL que descreva na tela a data de contratação e o salário dos empregados do departamento 60

```
SET SERVEROUTPUT ON
DECLARE
      emp_hiredate employees.hire_date%TYPE;
      emp_salary employees.salary%TYPE;
BEGIN
      SELECT hire_date, salary
      INTO emp_hiredate, emp_salary
      FROM employees
      WHERE department_id = 60;
      DBMS_OUTPUT.PUT_LINE ('data contratação = '
      || emp_hiredate || ' e salário = ' ||
      emp_salary);
END;
/
```

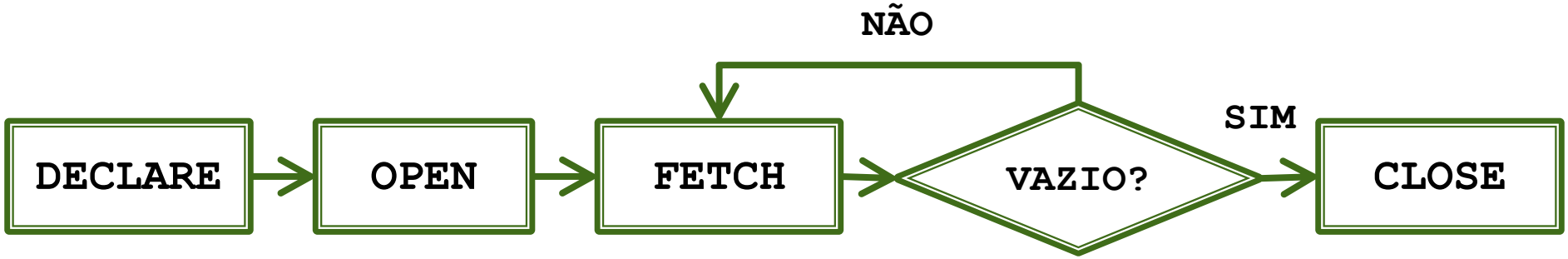


Observações – SELECT INTO

- Se um comando SELECT INTO retorna mais de uma tupla
  - Exceção TOO_MANY_ROWS
  - SQL%ROWCOUNT = 1, não contém o valor real de tuplas que satisfazem a consulta
- Se um comando SELECT INTO não retorna resultado
  - Exceção NO_DATA_FOUND
  - Interrompe imediatamente o fluxo de controle, antes mesmo de poder checar SQL%NOTFOUND
- Um comando SELECT INTO que chama uma função de agregação sempre retorna um valor ou NULL
  - SQL%NOTFOUND é sempre FALSE

Cursores explícitos

- Declarados e gerenciados explicitamente pelo programador
- Fluxo de controle básico de cursores explícitos



Cria um cursor na memória

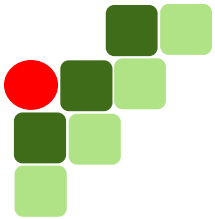
Abre o cursor, coloca os dados na memória e aponta para a primeira tupla

Carrega a tupla corrente e aponta para a próxima

Verifica tuplas existentes

Retorna a FETCH se houver mais tuplas

Libera a área alocada na memória

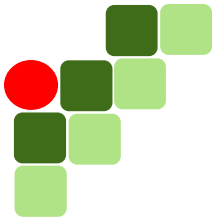


Declarando um cursor

```
CURSOR cursor_name IS
      select_statement;
```

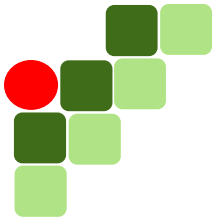
```
DECLARE
   CURSOR emp_cursor IS
        SELECT employee_id, hire_date, salary
        FROM employees
        WHERE department_id = 60;
```

```
DECLARE
    dept_id NUMBER := &dept_id;
    CURSOR emp_cursor2 IS
        SELECT * FROM employees
        WHERE department_id = dept_id;
```



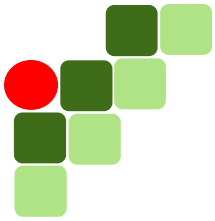
Abrindo um cursor

- Aloca recursos do SGBD para processar a consulta
- Processa a consulta, ou seja, identifica o conjunto de resultados que satisfaz às condições da consulta
- Posiciona o cursor na primeira linha do conjunto de resultados



Abrindo um cursor

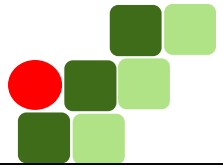
```
DECLARE
   CURSOR emp_cursor IS
        SELECT employee_id, hire_date, salary
        FROM employees
        WHERE department_id = 60;
...
BEGIN
   OPEN emp_cursor;
   ...
```



Carregando dados do cursor

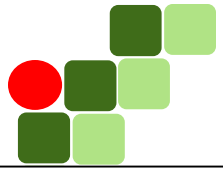
```
FETCH cursor_name INTO into_clause;
```

- Recupera a tupla corrente do conjunto de resultado
- Armazena os valores dos atributos da tupla corrente nas variáveis ou no registro usado em `into_clause`
- Avança o cursor para a próxima tupla



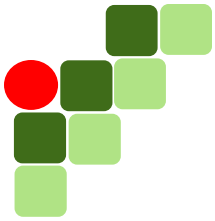
Carregando dados do cursor

```
DECLARE
    CURSOR emp_cursor IS
         SELECT employee_id, hire_date, salary
         FROM employees WHERE department_id = 60;
    empno employees.employee_id%TYPE;
    hdate employees.hire_date%TYPE;
    sal employees.salary%TYPE;
    ...
BEGIN
    OPEN emp_cursor;
    FETCH emp_cursor INTO empno, hdate, sal;
    DBMS_OUTPUT.PUT_LINE(empno ||' '|| hdate ||' '|| sal);
    ...
END;
/
```



Carregando dados do cursor

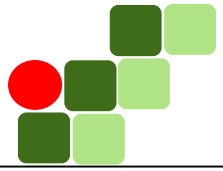
```
DECLARE
    CURSOR emp_cursor IS
         SELECT employee_id, hire_date, salary
         FROM employees WHERE department_id = 60;
  ...
BEGIN
   OPEN emp_cursor;
  LOOP
     FETCH emp_cursor INTO empno, hdate, sal;
     EXIT WHEN emp_cursor%NOTFOUND;
     DBMS_OUTPUT.PUT_LINE(empno ||' '|| hdate ||' '|| sal);
   END LOOP;
   ...
END;
/
```



Fechando um cursor

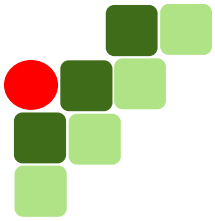
- Libera recursos de memória para serem reutilizados
- Depois de fechar um cursor, não é possível carregar mais tuplas ou referenciar seus atributos

```
  ...
 LOOP
    FETCH emp_cursor INTO empno, hdate, sal;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(empno ||' '|| hdate ||' '|| sal);
 END LOOP;
 CLOSE emp_cursor;
END;
/
```



Cursores e registros

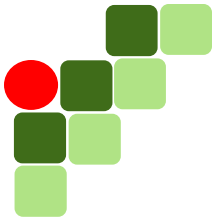
```
DECLARE
   CURSOR emp_cursor IS
     SELECT employee_id, last_name FROM employees
     WHERE department_id = 60;
   emp_record emp_cursor%ROWTYPE;
BEGIN
   OPEN emp_cursor;
   LOOP
     FETCH emp_cursor INTO emp_record;
     EXIT WHEN emp_cursor%NOTFOUND;
     DBMS_OUTPUT.PUT_LINE(emp_record.employee_id ||
                          ' ' || emp_record.last_name);
   END LOOP;
   CLOSE emp_cursor;
END;
```



Atributos de cursores explícitos

Atributo	Tipo	Descrição
%ISOPEN	boolean	Retorna TRUE se cursor está aberto
%FOUND*	boolean	Retorna TRUE se o FETCH mais recente encontrou uma tupla
%NOTFOUND*	boolean	Retorna TRUE se o FETCH mais recente não encontrou uma tupla (complemento de %FOUND)
%ROWCOUNT*	number	Retorna o número de tuplas encontradas até o momento

***somente possuem valor depois do primeiro fetch (por isso utilize o comando loop... exit when)**

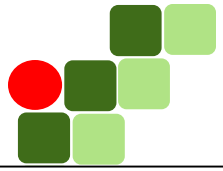


Atributo %ISOPEN

- Use %ISOPEN antes de realizar um FETCH para verificar se o cursor está aberto

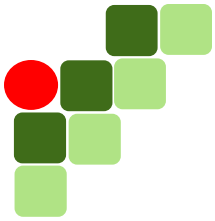
```
IF NOT emp_cursor%ISOPEN THEN
      OPEN emp_cursor;
END IF;
LOOP
      FETCH emp_cursor ...
```

- Ao referenciar um cursor explícito que não está aberto, é lançada a exceção INVALID_CURSOR



Atributos de cursores explícitos

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name FROM employees;
  emp_record emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%ROWCOUNT > 10
              OR emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(emp_cursor%ROWCOUNT ||
                         ' ' || emp_record.last_name);
  END LOOP;
  CLOSE emp_cursor;
END;
```

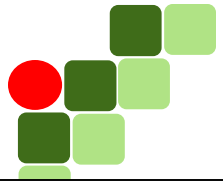


Cursor com parâmetros

- Permite passar valores de parâmetros para um cursor quando ele é aberto e a consulta é executada
- É possível abrir um cursor explícito diversas vezes com conjunto de respostas diferentes

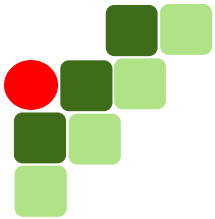
```
CURSOR nome_cursor
[(parametro1 datatype, parametro2 datatype, ...)]
IS
      comando_select;
```

```
OPEN nome_cursor(valor_parametro1,valor_parametro2, ...);
```



Cursor com parâmetros

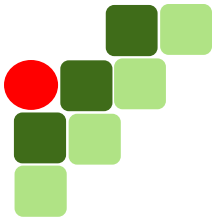
```
DECLARE
    CURSOR emp_cursor (dept NUMBER) IS
        SELECT employee_id, last_name
        FROM employees
        WHERE department_id = dept;
BEGIN
    OPEN emp_cursor(60);
    ...
    CLOSE emp_cursor;
    OPEN emp_cursor(80);
    ...
    CLOSE emp_cursor;
    ...
END;
/
```



Exercícios

- Faça um bloco anônimo que tenha duas variáveis de substituição (leitura no prompt): função e salário máximo permitido
  - Imprime na tela o nome e o sobrenome dos empregados da função especificada que ganham mais que o salário máximo permitido
  - Imprime também o quanto ganham a mais
- Faça os testes para:
  - `'ST_CLERK', 3000`
  - `'SA_REP', 10000`

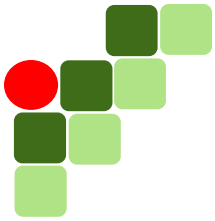
```
DECLARE
  jobid HR.EMPLOYEES.JOB_ID%TYPE;
  maxsal HR.EMPLOYEES.SALARY%TYPE;
  CURSOR overpaid (j VARCHAR2, max_sal NUMBER) IS
      SELECT last_name, first_name, (salary - max_sal) overpay
      FROM employees WHERE job_id = j AND salary > max_sal
      ORDER BY salary;
    emp_record overpaid%ROWTYPE;
BEGIN
    jobid := '&jobid';
    maxsal := '&maxsal';
    OPEN overpaid(jobid, maxsal);
    LOOP
      FETCH overpaid INTO emp_record;
      EXIT WHEN overpaid%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(emp_record.last_name || ', ' ||
emp_record.first_name || ' (excedente = ' || emp_record.overpay ||
')');
    END LOOP;
    CLOSE overpaid;
END;
```



Cursor FOR Loops

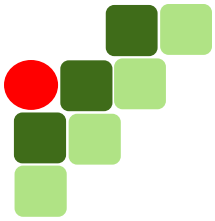
- O cursor FOR Loop simplifica o processamento de cursores explícitos
- Implicitamente abre, carrega, verifica e fecha o cursor
- O registro é implicitamente declarado

```
FOR nome_registro IN nome_cursor LOOP
      comando1;
      comando2;
      . . .
END LOOP;
```



Cursor FOR Loops

```
DECLARE
  CURSOR emp_cursor IS
    SELECT employee_id, last_name FROM employees
    WHERE department_id = 60;
BEGIN
  FOR emp_record IN emp_cursor LOOP
     DBMS_OUTPUT.PUT_LINE(emp_record.employee_id ||
                          ' ' || emp_record.last_name);
  END LOOP;
END;
/
```

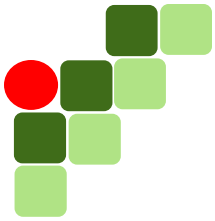


Exercício

- Faça um bloco anônimo que imprima na tela o sobrenome, função (job_id) e salário de todos os empregados. Para os empregados que recebem algum valor de comissão, mostre o salário incluindo a porcentagem de comissão
  - Para implementar o bloco anônimo, use o comando FOR... IN... LOOP

```
FOR nome_registro IN nome_cursor LOOP
      comando1;
      comando2;
      . . .
END LOOP;
```

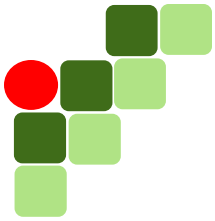
```
DECLARE
  CURSOR emp_cursor IS
    SELECT last_name, job_id, salary, commission_pct
    FROM employees;
BEGIN
  FOR emp_record IN emp_cursor LOOP
     DBMS_OUTPUT.PUT(emp_record.last_name || ' ' ||
emp_record.job_id || ' ');
     IF emp_record.commission_pct IS NOT NULL THEN
         DBMS_OUTPUT.PUT_LINE(emp_record.salary || ' '
|| TO_CHAR(emp_record.salary + emp_record.salary *
emp_record.commission_pct));
     ELSE
         DBMS_OUTPUT.PUT_LINE(emp_record.salary);
     END IF;
  END LOOP;
END;
/
```



Cursor FOR Loops usando subconsulta

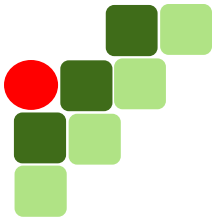
- Não é necessário declarar o cursor

```
BEGIN
  FOR emp_record IN
          (SELECT employee_id, last_name
           FROM employees WHERE department_id = 60)
  LOOP
     DBMS_OUTPUT.PUT_LINE(emp_record.employee_id
                          ||' '|| emp_record.last_name);
  END LOOP;
END;
/
```



Exercícios

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

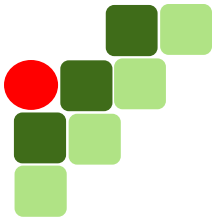


Exercícios

- Faça um bloco anônimo que calcule o tempo de empresa (em anos) de todos empregados e mostre na tela seu nível:
  - Sênior: acima de 15 anos de empresa
  - Pleno: entre 5 e 15 anos
  - Júnior: menor que 5 anos

- Sugestão: usar a função

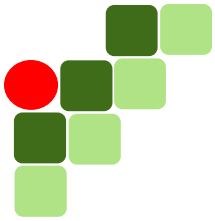
```
MONTHS_BETWEEN(SYSDATE, date)/12;
```



Exercícios

- Criar uma tabela retired_emps contendo as colunas: employee_id, last_name, job_id, hire_date, retire_date, salary, department_id

- Criar um bloco anônimo PL/SQL que verifique todos empregados e os insira na tabela retired_emps caso tenham mais de 18 anos de empresa
  - Use registros



Referências

- Oracle Database PL/SQL Language Reference
  - Implicit Cursors
  - Explicit Cursors