



CESUMAR – CENTRO UNIVERSITÁRIO DE MARINGÁ

Banco de Dados

Aparecido Vilela Junior



CURSOR



CURSOR

- Sabemos que o resultado de uma operação relacional é sempre uma tabela composta de zero, uma ou mais rows, de acordo com o critério de restrição informado.
- Um cursor representa uma tabela temporariamente armazenada em memória e criada como resultado dos comandos: select, insert, update, delete, commit, rollback.
- Ele contém os registros afetados pelo comando que provocou sua criação



CURSOR

- Implícito:
 - Dispensa qualquer tipo de tratamento

- Explícito:
 - Gerado apenas pelo select, que deve ser declarado e manipulado via comandos próprios.

- Criando um cursor Explícito, podemos carregar o conteúdo de uma tabela em memória sem precisar emitir tantos select/into quantos forem os registros da tabela



CURSOR

- Na declaração de um cursor, já é feita a associação com o comando Select capaz de obter as linhas do banco de dados.
- O cursor deve ser criado antes de ser usado, e por isso é necessário que a declaração esteja presente no bloco em que usaremos o cursor ou em um bloco mais externo



CURSOR - Declaração

- O tipo de retorno deve representar um registro correspondendo às colunas trazidas pelo comando Select
- A manipulação de um cursor é semelhante àquela usada para arquivos convencionais:
 - ☐ Abrir (Open)
 - ☐ Ler uma linha (Fetch Into)
 - ☐ Fechar (Close)



CURSOR - Declaração

■ Ex:

```
DECLARE
```

```
  CURSOR C1 IS SELECT DEPTNO FROM DEPT;
```

```
  CURSOR C2 (PDEPTO IN NUMBER := 10) IS
```

```
    SELECT COUNT(*), MAX(SAL)
```

```
      FROM EMP
```

```
      WHERE DEPTNO = PDEPTO;
```

```
  CURSOR C3 (PDEPTO IN NUMBER := 20)
```

```
    RETURN EMP%ROWTYPE IS
```

```
      SELECT * FROM EMP
```

```
      WHERE DEPTNO = PDEPTO
```

```
      ORDER BY SAL DESC;
```

```
BEGIN
```

```
  NULL;
```

```
END;
```



CURSOR

- Há quatro atributos que podem ser utilizados tanto por cursores implícitos quanto explícitos:
 - SQL%ROWCOUNT: Informa quantas linhas foram afetadas pelo comando que gerou o cursor
 - SQL%FOUND: Será **true** caso alguma linha tenha sido afetada
 - SQL%NOTFOUND: Será **false** caso alguma linha tenha sido afetada
 - SQL%ISOPEN: Será **true** caso o cursor esteja aberto. Somente faz sentido para cursores explícitos.



CURSOR - OPEN

- A abertura de um cursor é realizada com o comando Open.
- Os parâmetros devem ser fornecidos a tempo de Open para que seja possível a execução de um comando Select
- Os parâmetros declarados com valor inicial podem, a tempo de Open, não receber nenhum valor e utilizar os valores iniciais previamente declarados



CURSOR - OPEN

- Para a passagem dos parâmetros, podemos usar a notação posicionada ou a notação nomeada;
- Notação **Nomeada**: Quando utilizamos o nome do parâmetro e o símbolo “=>”, podemos passar os parâmetros em qualquer ordem



CURSOR - OPEN

DECLARE

CURSOR C1 IS SELECT DEPTNO FROM DEPT;

CURSOR C2 (PDEPTO IN NUMBER := 10) IS

SELECT COUNT(*), MAX(SAL)

FROM EMP

WHERE DEPTNO = PDEPTO;

CURSOR C3 (PDEPTO IN NUMBER := 20)

RETURN EMP%ROWTYPE IS

SELECT * FROM EMP

WHERE DEPTNO = PDEPTO

ORDER BY SAL DESC;

CURSOR C4 (P1 IN VARCHAR2, P2 IN NUMBER) IS

SELECT * FROM EMP

WHERE JOB = P1 AND DEPTNO = P2;

BEGIN

OPEN C1;

OPEN C2;

OPEN C3 (30);

OPEN C4 (P2 => 40, P1 => 'SALESMAN');

END;



CURSOR - FETCH

- Associa os valores da linha atual às variáveis do programa e, posteriormente, posiciona o cursor na próxima linha do conjunto de linhas resultantes da operação de Select.
- As variáveis devem ser informadas no Fetch na mesma ordem das expressões correspondentes no comando Select.



CURSOR - FETCH

- Para cada expressão, deve haver uma variável receptora.
- O tipo de dado da variável e da expressão deve ser compatível.



CURSOR - FETCH

```
DECLARE
CURSOR C1 IS SELECT DEPTNO FROM DEPT;
CURSOR C2 (PDEPTO IN NUMBER := 10) IS
    SELECT COUNT(*), MAX(SAL) FROM EMP
    WHERE DEPTNO = PDEPTO;
CURSOR C3 (PDEPTO IN NUMBER := 20)
    RETURN EMP%ROWTYPE IS
    SELECT * FROM EMP
    WHERE DEPTNO = PDEPTO
    ORDER BY SAL DESC;
CURSOR C4 (P1 IN VARCHAR2, P2 IN NUMBER) IS
    SELECT * FROM EMP
    WHERE JOB = P1 AND DEPTNO = P2;
VCD_DEPTO DEPT.DEPTNO%TYPE;
VQTD    NUMBER;
VMAX    NUMBER;
BEGIN
    OPEN C1;
    <<INICIO>>
        FETCH C1 INTO VCD_DEPTO;
        OPEN C2(VCD_DEPTO);
        FETCH C2 INTO VQTD, VMAX;
        OPEN C3 (VCD_DEPTO);
        OPEN C4 (P2 => 40, P1 => 'SALESMAN');
    <<LOOP>>
        NULL;
    END;
```



CURSOR - %ROWTYPE

- A fim de facilitar a recepção de informações das linhas lidas a PL/SQL implementa o atributo %ROWTYPE.
- O atributo gera uma área (registro) capaz de armazenar, exatamente, uma linha com o layout da linha de uma tabela do banco de dados.
- Ou com o layout apenas das colunas (ou expressões) referenciadas no Select associado a um cursor



CURSOR - %ROWTYPE

DECLARE

CURSOR C1 IS SELECT DEPTNO FROM DEPT;

CURSOR C2 (PDEPTO IN NUMBER := 10) IS

SELECT COUNT(*) QTDE, MAX(SAL) FROM EMP WHERE DEPTNO = PDEPTO;

CURSOR C3 (PDEPTO IN NUMBER := 20)

RETURN EMP%ROWTYPE IS

SELECT * FROM EMP

WHERE DEPTNO = PDEPTO

ORDER BY SAL DESC;

CURSOR C4 (P1 IN VARCHAR2, P2 IN NUMBER) IS

SELECT * FROM EMP

WHERE JOB = P1 AND DEPTNO = P2;

VCD_DEPTO DEPT.DEPTNO%TYPE;

RTOTAL C2%ROWTYPE;

RFUNC EMP%ROWTYPE;

BEGIN

OPEN C1;

<<INICIO>>

FETCH C1 INTO VCD_DEPTO;

OPEN C2(VCD_DEPTO);

FETCH C2 INTO RTOTAL;

DBMS_OUTPUT.PUT_LINE('A quantidade de linhas é '||rtotal.QTDE);

OPEN C3 (VCD_DEPTO);

OPEN C4 (P2 => 40, P1 => 'SALESMAN');

<<LOOP>>

FETCH C3 INTO RFUNC;

END;



CURSOR - %ROWTYPE

- ❑ DECLARE
- ❑ CURSOR C1 IS SELECT SAL * 1.2 PREVISAO
- ❑ FROM EMP;
- ❑ RC1 C1%ROWTYPE;
- ❑ MSG VARCHAR2(200);
- ❑ BEGIN
- ❑ IF RC1.PREVISAO > 500000 THEN
- ❑ MSG := 'TOTAL ACIMA DE 500000';
- ❑ ELSE
- ❑ MSG := 'AUMENTO APROVADO';
- ❑ END IF;
- ❑ DBMS_OUTPUT.PUT_LINE(MSG);
- ❑ END;



CURSOR - CLOSE

- Para fecharmos um cursor, devemos utilizar o comando CLOSE.
- Esse comando libera a área reservada para a montagem do conjunto de dados selecionados
- Sintaxe:
 - CLOSE <Nome do Cursor>



CURSORES IMPLÍCITOS

- Declarado implicitamente pelo Oracle.
- O Oracle implicitamente abre um cursor para cada comando de SQL que não tenha associado um cursor explícito, inclusive para SELECT INTO. Podemos referenciar este cursor usando a palavra SQL.
- Não podemos usar essa referência para executar os comandos Open, Fetch ou Close, porém podemos utilizá-la para obter informações sobre o último comando executado.



CURSOR - %FOUND

- É um atributo de cursor que indica se a última operação de **FETCH** foi bem sucedida (para cursores explícitos) ou se alguma linha foi afetada pelo último comando Insert, Update ou Delete.
- E se o comando Select Into retornou uma ou mais linhas (para cursores implícitos)



CURSOR - %FOUND

- ❑ DECLARE
- ❑ VCD_DEPTO NUMBER(2) := &DEPTO;
- ❑ CURSOR C1 IS SELECT * FROM EMP
- ❑ WHERE DEPTNO = VCD_DEPTO;
- ❑ RC1 C1%ROWTYPE;
- ❑ BEGIN
- ❑ DELETE FROM EMP WHERE EMPNO = 7521;
- ❑ IF SQL%FOUND THEN
- ❑ OPEN C1;
- ❑ <<LOOP>>
- ❑ FETCH C1 INTO RC1;
- ❑ IF C1%FOUND THEN
- ❑ DBMS_OUTPUT.PUT_LINE(RC1.EMPNO);
- ❑ GOTO LOOP;
- ❑ END IF;
- ❑ END IF;
- ❑ END;



CURSOR - %NOTFOUND

- É um atributo de cursor com a lógica inversa ao atributo %FOUND.
- O %NOTFOUND retornará false se o último comando Fetch retornar uma linha (para cursores explícitos) ou se o último comando Insert, Update, ou Delete não afetar nenhuma linha ou, ainda, se o último comando Select Into não retornar nenhuma linha (para cursores implícitos)



CURSOR - %ISOPEN

- Permite que verifique se um cursor está aberto ou não.
- Para cursores implícitos, o resultado será sempre False uma vez que o Oracle fecha o cursor imediatamente após a operação e antes de a ação retornar ao programa.



CURSOR - %ISOPEN

- DECLARE
- VCD_DEPTO NUMBER(02) := &DEPTO;
- CURSOR C1 IS SELECT * FROM EMP
- WHERE DEPTNO = VCD_DEPTO;
- RC1 C1%ROWTYPE;
- BEGIN
- OPEN C1;
- <<LOOP>>
- FETCH C1 INTO RC1;
- IF C1%NOTFOUND THEN
- GOTO FIM;
- END IF;
- DBMS_OUTPUT.PUT_LINE(RC1.EMPNO);
- GOTO LOOP;
- <<FIM>>
- IF C1%ISOPEN THEN
- CLOSE C1;
- END IF;
- END;



CURSOR - %ROWCOUNT

- Indica o número de linhas já lidas (fetched) para os cursores explícitos ou o número de linhas afetadas por um comando Insert, Update ou Delete, ou ainda, o número de linhas retornadas por um comando Select Into (para cursores implícitos)
- Quando o cursor é aberto, o atributo %RowCount é zerado.
- Antes do primeiro Fetch, o atributo continua com zero.
- O valor é incrementado após o Fetch ter sido efetuado com sucesso.
- Até que o primeiro comando de DML seja executado, o valor do atributo é Null pra cursores implícitos



CURSOR - %ROWCOUNT

- ❑ DECLARE
- ❑ VCD_DEPTO NUMBER(02) := &DEPTO;
- ❑ CURSOR C1 IS SELECT * FROM EMP
- ❑ WHERE DEPTNO = VCD_DEPTO;
- ❑ RC1 C1%ROWTYPE;
- ❑ BEGIN
- ❑ OPEN C1;
- ❑ <<LOOP>>
- ❑ FETCH C1 INTO RC1;
- ❑ IF C1%NOTFOUND OR C1%ROWCOUNT > 5 THEN
- ❑ GOTO FIM;
- ❑ END IF;
- ❑ DBMS_OUTPUT.PUT_LINE(RC1.EMPNO);
- ❑ GOTO LOOP;
- ❑ <<FIM>>
- ❑ IF C1%ISOPEN THEN
- ❑ CLOSE C1;
- ❑ END IF;
- ❑ END;



CURSOR – LOOP/EXIT

- DECLARE
- CURSOR C1 IS SELECT SAL, EMPNO FROM EMP;
- RC1 C1%ROWTYPE;
- MSG VARCHAR2(200);
- BEGIN
- OPEN C1;
- MSG := 'FuncionarioS = ';
- LOOP
- FETCH C1 INTO RC1;
- IF C1%FOUND THEN
- MSG := MSG||RC1.EMPNO||' - ';
- ELSE
- EXIT;
- END IF;
- END LOOP;
- DBMS_OUTPUT.PUT_LINE(MSG);
- END;



CURSOR - WHILE

- O comando WHILE corresponde a uma outra forma de Loop em que estabelecemos uma condição de interrupção do processamento.
- A cláusula WHEN no comando Exit oferece uma forma condicional de interrompermos o processamento.



CURSOR - WHILE

- DECLARE
- CURSOR C1 IS SELECT SAL, EMPNO FROM EMP
- ORDER BY SAL DESC;
- RC1 C1%ROWTYPE;
- MSG VARCHAR2(200);
- BEGIN
- OPEN C1;
- MSG := 'Funcionarios = ';
- FETCH C1 INTO RC1;
- WHILE C1%FOUND LOOP
- EXIT WHEN C1%ROWCOUNT > 10;
- MSG := MSG||RC1.EMPNO||' - ';
- FETCH C1 INTO RC1;
- END LOOP;
- DBMS_OUTPUT.PUT_LINE(MSG);
- END;



CURSOR – FOR LOOP

- O comando For Loop determina que a sequência de comandos seja executada um número fixo de vezes.
- O número de iterações é conhecido (determinado) antes de o loop ter início.
- Os valores referentes a valor inferior e valor superior podem ser fornecidos por constantes e variáveis



CURSOR – FOR LOOP

- DECLARE
- VEZES NUMBER := 0;
- INICIO NUMBER := '&INICIO';
- FIM NUMBER := '&FIM';
- MSG VARCHAR2(200);
- BEGIN
- FOR I IN INICIO..FIM LOOP
- VEZES := VEZES + 1;
- END LOOP;
- FOR I IN FIM..INICIO LOOP
- VEZES := VEZES + 1;
- END LOOP;
- dbms_output.PUT_LINE('O valor de VEZES é '|| VEZES);
- END;



CURSOR – FOR LOOP

- ❑ DECLARE
- ❑ VEZES NUMBER := 5;
- ❑ INICIO NUMBER := '&INICIO';
- ❑ FIM NUMBER := '&FIM';
- ❑ MSG VARCHAR2(200);
- ❑ BEGIN
- ❑ MSG := 'Valores de I: ';
- ❑ FOR I IN REVERSE 1..5 LOOP
- ❑ MSG := MSG ||I|| ' ';
- ❑ END LOOP;
- ❑ <<EXTERNO>>
- ❑ FOR I IN 3..15 LOOP
- ❑ <<INTERNO>>
- ❑ FOR J IN INICIO..FIM LOOP
- ❑ IF J > I THEN
- ❑ EXIT EXTERNO;
- ❑ END IF;
- ❑ END LOOP INTERNO;
- ❑ END LOOP EXTERNO;
- ❑ DBMS_OUTPUT.PUT_LINE(MSG);
- ❑ END;



CURSOR – CURSOR LOOP

- Este comando é similar ao comando For Loop, mas é específico para utilização com cursores.
- O cursor pode ser declarado previamente e utilizado no comando, ou pode ser incluído o comando Select desejado no próprio comando FOR



CURSOR - UPDATE

- Podemos selecionar uma determinada linha usando o cursor e em seguida atualizá-la, sem que seja necessária uma nova pesquisa.
- A cláusula WHERE CURRENT OF indica que será atualizada a linha atualmente apontada pelo cursor.
- Para que essa operação seja legal, devemos utilizar a cláusula FOR UPDATE no comando Select do cursor associado
- A cláusula FOR UPDATE adicionada ao comando Select indica que todas as linhas escolhidas através da condição presentes na cláusula WHERE devem sofrer bloqueio (Lock) contra atualizações correntes.



CURSOR - UPDATE

```
■ Ex:
DECLARE
  CURSOR C1      IS SELECT SAL, EMPNO FROM EMP
                  FOR UPDATE OF SAL;
  RC1            C1%ROWTYPE;
BEGIN
  OPEN C1;
  <<LOOP>>
    FETCH C1 INTO RC1;
    IF C1%FOUND THEN
      IF RC1.SAL < 3000 THEN
        UPDATE EMP SET SAL = SAL * 1.3
          WHERE CURRENT OF C1;
      END IF;
      GOTO LOOP;
    END IF;
  COMMIT;
END;
```



CURSOR - DELETE

- Da mesma forma que o Update, o comando Delete também possui uma sintaxe especial para trabalhar com cursores.
- Com isso, poderemos remover a linha apontada pelo cursor sem haver necessidade de estabelecermos uma nova condição de busca.



CURSOR - DELETE

```
DECLARE
  CURSOR C1    IS SELECT SAL, EMPNO FROM EMP
                FOR UPDATE OF SAL;
  RC1          C1%ROWTYPE;
BEGIN
  OPEN C1;
  <<LOOP>>
    FETCH C1 INTO RC1;
    IF C1%FOUND THEN
      IF RC1.SAL IS NULL THEN
        DELETE FROM EMP WHERE CURRENT OF C1;
      END IF;
      GOTO LOOP;
    END IF;
  COMMIT;
END;
```

■ /



CURSOR – CURSOR LOOP

- DECLARE
- CURSOR C1 IS SELECT * FROM EMP;
- VEZES NUMBER := 0;
- MSG VARCHAR2(200);
- BEGIN
- FOR RC1 IN C1 LOOP
- VEZES := VEZES + 1;
- END LOOP;
- MSG := 'VEZES = ' || VEZES;
- FOR RC1 IN (SELECT * FROM EMP) LOOP
- VEZES := VEZES + 1;
- END LOOP;
- DBMS_OUTPUT.PUT_LINE(MSG || ' - VEZES = ' || VEZES);
- END;
- /



Exercícios

- CREATE TABLE ALUNO (RA NUMBER(9), DISCIPLINA VARCHAR2(30), MEDIA NUMBER(3,1), CARGA_HORA NUMBER(2), FALTAS NUMBER(2), RESULTADO VARCHAR2(10));
- Inserir uma linha deixando a coluna RESULTADO em branco.
- INSERT INTO ALUNO VALUES (1,'DISC 1',7.5,80,20,");
- INSERT INTO ALUNO VALUES (2,'DISC 1',5.5,80,20,");
- INSERT INTO ALUNO VALUES (3,'DISC 1',7.5,80,40,");
- Criar um bloco PL/SQL para preencher a coluna resultado conforme o seguinte:
- Se o aluno obteve média igual ou maior que 7.0 e suas faltas não ultrapassarem 25% da carga horária da disciplina o resultado será: APROVADO.
- Se o aluno obteve média inferior a 7.0 e suas faltas não ultrapassarem 25% da carga horária da disciplina o resultado será: EXAME.
- Para demais casos o resultado será: REPROVADO.



Exercícios

- 1. Criar a tabela PRODUTO:
- `CREATE TABLE PRODUTO (CODIGO NUMBER(4), VALOR NUMBER(7,2));`
- Inserir os valores:
- `INSERT INTO PRODUTO VALUES (1000,300);`
- `INSERT INTO PRODUTO VALUES (1001,500);`
- `INSERT INTO PRODUTO VALUES (2000,300);`
- `INSERT INTO PRODUTO VALUES (2001,500);`
- Criar um bloco PL/SQL para atualizar os preços conforme segue:
 - Produtos com CODIGO inferior a 2000: Acrescentar 10% ao VALOR atual.
 - Produtos com CODIGO igual ou superior a 2000: Acrescentar 20% ao VALOR atual.



PRÁTICA

- 1) Leia os n maiores salários da tabela de funcionários e apresente-os em ordem ascendente
 - 2) Crie uma tabela com o comando apresentado abaixo para receber os n maiores salários da tabela de empregados. Na coluna total deve ser armazenado o somatório de todos os salários lidos
 -
 -
 -
 -
 -
 -
 -
 -
 -
- ```
CREATE TABLE TABTOTAL (
 NOME VARCHAR2(25),
 SALARIO NUMBER(08),
 TOTAL NUMBER(11))
```
- Inclua todos os dados e posteriormente atualize a coluna total com o valor obtido na soma.



# PRÁTICA - CONTINUAÇÃO

- 3) Crie uma tabela com o comando apresentado abaixo para receber a quantidade de funcionários por departamento. Utilize Loop e Exit When.
- CREATE TABLE RESULTADO (  
■           COD\_DEP   NUMBER,  
■           QTD\_EMP   NUMBER,  
■           VLR\_SAL   NUMBER,  
■           MED\_SAL   NUMBER);
- Neste exercício, só deve ser preenchida a quantidade de funcionários por departamento.
  
- 4) Utilizando a mesma tabela anterior, já parcialmente preenchida, complete-a com o total de salários e a média salarial. Utilize Cursor Loop, Inicie a leitura pela tabela resultado e trabalhe com Update Where Current.



# Resolução – Exercício 01

- DECLARE
- QTDE NUMBER := &QT;
- MSG VARCHAR2 (200);
- CURSOR C1 IS SELECT SAL FROM EMP
- WHERE SAL IS NOT NULL
- ORDER BY SAL DESC;
- BEGIN
- MSG := '';
- FOR RC1 IN C1 LOOP
- IF QTDE > 0 THEN
- QTDE := QTDE - 1;
- ELSE
- EXIT;
- END IF;
- MSG := RC1.SAL || ' ' || MSG;
- END LOOP;
- DBMS\_OUTPUT.PUT\_LINE( 'Os maiores salarios sao : ' || MSG);
- END;
- /



# Resolução – Exercício 02

- DECLARE
- QTDE NUMBER := NVL(TO\_NUMBER('&QTD'),0);
- VTOTAL NUMBER := 0;
- CURSOR C1 IS SELECT ENAME, SAL FROM EMP
- WHERE SAL IS NOT NULL
- ORDER BY SAL DESC;
- BEGIN
- DELETE FROM TABTOTAL;
- FOR RC1 IN C1 LOOP
- EXIT WHEN C1%ROWCOUNT > QTDE;
- VTOTAL := VTOTAL + RC1.SAL;
- INSERT INTO TABTOTAL VALUES (RC1.ENAME, RC1.SAL,NULL);
- END LOOP;
- UPDATE TABTOTAL
- SET TOTAL = VTOTAL;
- COMMIT;
- END;



# Resolução – Exercício 03

- DECLARE
- CODIGO NUMBER;
- CURSOR C1 IS SELECT DEPTNO, COUNT(EMPNO) QTDE FROM EMP
- WHERE DEPTNO IS NOT NULL
- GROUP BY DEPTNO;
- RC1 C1%ROWTYPE;
- BEGIN
- OPEN C1;
- LOOP
- FETCH C1 INTO RC1;
- EXIT WHEN C1%NOTFOUND;
- CODIGO := RC1.DEPTNO;
- INSERT INTO RESULTADO (COD\_DEP, QTD\_EMP) VALUES (CODIGO,RC1.QTDE);
- END LOOP;
- END;
- /



# Resolução – Exercício 04

```
■ DECLARE
■ CURSOR C1 IS SELECT * FROM RESULTADO
■ FOR UPDATE OF VLR_SAL, MED_SAL;
■ V_DEP NUMBER;
■ V_SOMA NUMBER;
■ V_MEDIA NUMBER;
■ BEGIN
■ FOR RC1 IN C1 LOOP
■ V_DEP := RC1.COD_DEP;
■ SELECT ROUND(SUM(SAL)), ROUND (AVG(SAL))
■ INTO V_SOMA, V_MEDIA
■ FROM EMP
■ WHERE DEPTNO = V_DEP;
■ DBMS_OUTPUT.PUT_LINE(V_DEP || ' - ' || V_SOMA || ' - ' || V_MEDIA) ;
■ UPDATE RESULTADO
■ SET VLR_SAL = V_SOMA,
■ MED_SAL = V_MEDIA
■ WHERE CURRENT OF C1;
■ END LOOP;
■ END;
■ /
```