游标提供了一种从表中检索数据并进行操作的灵活手段，游标主要用在服务器上，处理由客户端发送给服务器端的SQL语句，或是批处理、存储过程、触发器中的数据处理请求。游标的作用相当于指针，通过游标PL/SQL程序可以一次处理查询结果集中的一行，并可以对该行数据执行特定操作，从而为用户在处理数据的过程中提供了很大方便。

在Oracle中，通过游标操作数据主要使用显式游标和隐式游标，另外，还有具有引用类型的ref游标，下面将分别介绍。

1 显式游标

1.1 显式游标介绍

显式游标由用户自己定义和操作游标，通常所说的游标都是指显式游标。
游标的使用分成以下4个步骤：

1.1.1 声明游标

在DECLARE部分按以下格式声明游标：
CURSOR 游标名[(参数1 数据类型[,参数2 数据类型...])]
IS SELECT 语句;
参数是可选部分，所定义的参数可以出现在SELECT语句的WHERE子句中。如果定义了参数，则必须在打开游标时传递相应的实际参数。

1.1.2 打开游标

在可执行部分，按以下格式打开游标：
OPEN 游标名[(实际参数1[,实际参数2])];
打开游标时，SELECT语句的查询结果就被传送到了游标工作区。

1.1.3 提取数据

在可执行部分，按以下格式将游标工作区中的数据提取到变量中。提取操作必须在打开游标之后进行。
FETCH 游标名 INTO 变量名1[,变量名2...];

或
FETCH 游标名 INTO 记录变量;
游标打开后有一个指针指向数据区，FETCH语句一次返回指针所指的一样数据，要返回多行需重复执行，可以使用循环语句来实现。控制循环可以通过判断游标的属性来进行。
对以上两种格式进行说明：
第一种格式中的变量是用来从游标中接收数据的变量，需要事先定义。变量的个数与类型应与SELECT语句中的字段变量的个数与类型一致。
第二种格式一次将一行数据取到记录变量中，需使用%ROWTYPE事先定义记录变量，这种形式使用起来比较方便，不必分别定义和使用多个变量。（定义记录变量方法：变量名 表名|游标名%ROWTYPE)

1.1.4 关闭游标

CLOSE 游标名;
显示游标打开后，必须显示地关闭。游标一旦关闭，游标占用的资源就被释放，游标变成无效，必须重新打开才能使用。

1.2 显式游标属性

通过游标的属性可以获取SQL的执行结果以及游标的状态信息：

- %found：布尔型，最近的FETCH语句返回一行数据则为真，否则为假；
- %notfound：布尔型，与%found属性相反；
- %rowcount：整数，获得FETCH语句返回的行数；
- %isopen：布尔型，游标已经打开时值为真，否则为假；

1.2.1 示例

```
DECLARE
  CURSOR cur_emp IS
    SELECT *
      FROM (SELECT empno, ename, dname, sal
              FROM scott.emp a, scott.dept b
             WHERE a.deptno = b.deptno
             ORDER BY sal DESC)
     WHERE rownum < 4;
```

```
  v_emp cur_emp%ROWTYPE;
BEGIN
  OPEN cur_emp;
  LOOP
    FETCH cur_emp
      INTO v_emp;
    EXIT WHEN cur_emp%NOTFOUND;

    dbms_output.put_line('EmpNO. :' || v_emp.empno || ' EName :' ||
                         v_emp.ename || ' DName :' || v_emp.dname ||
                         ' Sal :' || v_emp.sal);
  END LOOP;
  CLOSE cur_emp;
END;
```

1.3 游标参数传递

1.3.1 在条件中传递

```
DECLARE
  v_empno NUMBER := 7369;

  CURSOR cur_emp IS
    SELECT empno, ename, job FROM scott.emp WHERE empno = v_empno;

  v_emp cur_emp%ROWTYPE;
BEGIN
  OPEN cur_emp;
  LOOP
    FETCH cur_emp
      INTO v_emp;
    EXIT WHEN cur_emp%NOTFOUND;

    dbms_output.put_line('EmpNO. :' || v_emp.empno || ' EName :' ||
```

```
                    v_emp.ename || ' Job :' || v_emp.job);
  END LOOP;
  CLOSE cur_emp;
END;
```

1.3.2 在游标命名时传递

```
DECLARE
  CURSOR cur_emp(v_empno NUMBER) IS
    SELECT empno, ename, job FROM scott.emp WHERE empno = v_empno;

  v_emp cur_emp%ROWTYPE;
BEGIN
  OPEN cur_emp(7369);
  LOOP
    FETCH cur_emp
      INTO v_emp;
    EXIT WHEN cur_emp%NOTFOUND;

    dbms_output.put_line('EmpNO. :' || v_emp.empno || ' EName :' ||
                    v_emp.ename || ' Job :' || v_emp.job);
  END LOOP;
  CLOSE cur_emp;
END;
```

2 隐式游标

2.1 隐式游标介绍

在执行一个SQL语句时，Oracle会自动创建一个隐式游标，这个游标是内存中处理该语句的工作区域，隐式游标主要是DML语句的执行结果。

2.2 隐式游标属性

隐式游标和显示游标属性相似，只不过在其前面加上SQL：

- sql%found：布尔型，值为TRUE代表插入、删除、更新或单行查询操作成功;
- sql%notfound：布尔型，与sql%found属性相反；
- sql%rowcount：整数，代表DML语句成功执行的行数;
- sql%isopen：布尔型，DML执行过程中为真，结束后为假;

2.2.1 实例

2.2.1.1 DML操作中的隐式游标

```
BEGIN
  UPDATE scott.emp SET sal = sal * 1.2 WHERE empno = 7369;

  IF SQL%FOUND THEN
    dbms_output.put_line('Update success .');
    COMMIT;
  ELSE
    dbms_output.put_line('Update fail .');
  END IF;
END;
```

2.2.1.2 for语句循环游标

```
BEGIN
  FOR rec IN (SELECT * FROM scott.emp) LOOP
    dbms_output.put_line('EmpNO.:' || rec.empno || ' EName:' || rec.ename ||
                         ' Job:' || rec.job);
  END LOOP;
END;
```

3 动态游标

3.1 动态游标介绍

定义游标类型语句：
TYPE 游标类型名 REF CURSOR;
声明游标变量语句：
游标变量名游标类型名;
在可执行部分可以如下形式打开一个动态游标：
OPEN 游标变量名 FOR 查询语句字符串;

3.2 示例

```
DECLARE
  TYPE cur IS REF CURSOR;
  v_cur cur;

  v_empno scott.emp.empno%TYPE;
  v_ename scott.emp.ename%TYPE;
  v_job   scott.emp.job%TYPE;

  v_sql VARCHAR2(1000);
BEGIN
  v_sql := 'SELECT empno,ename,job  FROM scott.emp where empno in(7369,7499)';

  OPEN v_cur FOR v_sql;

  LOOP
    FETCH v_cur
      INTO v_empno, v_ename, v_job;
    EXIT WHEN v_cur%NOTFOUND;
    dbms_output.put_line('EmpNO.:' || v_empno || ' EName:' || v_ename ||
                         ' Job:' || v_job);
  END LOOP;

  CLOSE v_cur;
END;
```