Exacto: un **REF CURSOR** se emplea precisamente para devolver al llamante un **conjunto de filas** (un “result set”) que luego puedes recorrer. A continuación algunos puntos clave:

1. **¿Para qué sirve?**
   * Permite que un procedimiento o función PL/SQL construya dinámicamente una consulta y entregue su resultado al bloque que lo invocó (o incluso a aplicaciones externas, como Java o .NET).
   * Es la forma estándar en Oracle de “devolver tablas” desde código PL/SQL.
2. **Ventajas frente a un cursor estático**
   * **Dinámico**: puedes adaptar el SQL en tiempo de ejecución (añadir filtros, diferentes columnas, uniones…).
   * **Flexible**: un mismo procedimiento puede abrir distintos REF CURSOR, según parámetros que reciba.
3. **Flujo típico**
   * El procedimiento recibe un parámetro IN OUT SYS\_REFCURSOR.
   * Dentro del procedimiento hace OPEN <cursor> FOR SELECT ….
   * El bloque que llamó al procedimiento obtiene ya el cursor abierto.
   * Ese bloque hace FETCH … INTO … en un bucle para leer fila a fila.
   * Finalmente cierra el cursor (CLOSE).
4. **Dónde puede usarse**
   * **PL/SQL puro**: como hemos visto, para intercambiar datos entre procedimientos internos.
   * **APIs externas**: desde JDBC, ODP.NET, ODBC… capturas el REF CURSOR como un ResultSet o similar.
5. **Alternativa moderna**
   * En Oracle 12c+ puedes usar **PL/SQL pipelined functions** para devolver colecciones como tablas, pero los REF CURSOR siguen siendo muy comunes y soportados en todas las versiones.

**Conclusión**: Sí, un REF CURSOR te devuelve un conjunto de filas (un result set) que luego recorres con FETCH. Es la forma principal en Oracle PL/SQL de “devolver tablas” desde procedimientos.

**1. REF CURSOR en PROCEDIMIENTO (sin paquete)**

-- 1. Creamos el procedimiento standalone

CREATE OR REPLACE PROCEDURE proc\_emp\_by\_dept (

p\_deptno IN NUMBER, -- Parámetro IN: número de departamento

p\_emp\_refcur IN OUT SYS\_REFCURSOR -- Parámetro IN OUT: cursor de salida

)

IS

BEGIN

-- 2. Abrimos el REF CURSOR con la consulta

OPEN p\_emp\_refcur FOR

SELECT empno, ename, sal -- Columnas a devolver

FROM emp -- Tabla de empleados

WHERE deptno = p\_deptno; -- Filtrado por deptno recibido

END proc\_emp\_by\_dept;

/

Uso en bloque anónimo:

DECLARE

v\_cur SYS\_REFCURSOR; -- 1. Declaro la variable cursor

v\_empno emp.empno%TYPE; -- 2. Variable para empno

v\_ename emp.ename%TYPE; -- 3. Variable para ename

v\_sal emp.sal%TYPE; -- 4. Variable para sal

BEGIN

-- 5. Invoco el procedimiento, pasando deptno=30

proc\_emp\_by\_dept(p\_deptno => 30,

p\_emp\_refcur => v\_cur);

-- 6. Recorro el cursor

LOOP

FETCH v\_cur INTO v\_empno, v\_ename, v\_sal; -- Trae una fila

EXIT WHEN v\_cur%NOTFOUND; -- Sale si no hay más datos

DBMS\_OUTPUT.PUT\_LINE(

v\_empno || ' - ' || v\_ename || ' - ' || v\_sal

); -- Muestra datos

END LOOP;

-- 7. Cierra el cursor al acabar

CLOSE v\_cur;

END;

/

**2. REF CURSOR en PROCEDIMIENTO (dentro de un paquete)**

-- 1. Definición del paquete

CREATE OR REPLACE PACKAGE pkg\_emp AS

PROCEDURE get\_by\_dept(

p\_deptno IN NUMBER,

p\_emp\_refcur IN OUT SYS\_REFCURSOR

);

END pkg\_emp;

/

-- 2. Cuerpo del paquete

CREATE OR REPLACE PACKAGE BODY pkg\_emp AS

PROCEDURE get\_by\_dept(

p\_deptno IN NUMBER,

p\_emp\_refcur IN OUT SYS\_REFCURSOR

)

IS

BEGIN

-- 3. Abre el cursor con la consulta filtrada

OPEN p\_emp\_refcur FOR

SELECT empno, ename, sal

FROM emp

WHERE deptno = p\_deptno;

END get\_by\_dept;

END pkg\_emp;

/

Uso:

DECLARE

c SYS\_REFCURSOR; -- 1. Variable cursor

e\_no emp.empno%TYPE; -- 2. empno

e\_nm emp.ename%TYPE; -- 3. ename

e\_sal emp.sal%TYPE; -- 4. sal

BEGIN

-- 5. Llamada al procedimiento del paquete

pkg\_emp.get\_by\_dept(30, c);

-- 6. Loop de lectura

LOOP

FETCH c INTO e\_no, e\_nm, e\_sal;

EXIT WHEN c%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(e\_no || ' ' || e\_nm || ' ' || e\_sal);

END LOOP;

-- 7. Cierre de cursor

CLOSE c;

END;

/

**3. REF CURSOR en FUNCIÓN (sin paquete)**

-- 1. Función que devuelve SYS\_REFCURSOR

CREATE OR REPLACE FUNCTION fn\_emp\_by\_dept (

p\_deptno IN NUMBER -- Parámetro IN: deptno

) RETURN SYS\_REFCURSOR -- Devuelve un REF CURSOR

IS

l\_cur SYS\_REFCURSOR; -- 2. Variable local de cursor

BEGIN

-- 3. Abre el cursor con la consulta

OPEN l\_cur FOR

SELECT empno, ename, sal

FROM emp

WHERE deptno = p\_deptno;

RETURN l\_cur; -- 4. Devuelve el cursor abierto

END fn\_emp\_by\_dept;

/

Uso:

DECLARE

c SYS\_REFCURSOR; -- 1. Cursor que recibirá el resultado

eno emp.empno%TYPE; -- 2. empno

enm emp.ename%TYPE; -- 3. ename

sl emp.sal%TYPE; -- 4. sal

BEGIN

-- 5. Asigno el cursor devuelto por la función

c := fn\_emp\_by\_dept(20);

-- 6. Lectura en bucle

LOOP

FETCH c INTO eno, enm, sl;

EXIT WHEN c%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(eno || '|' || enm || '|' || sl);

END LOOP;

-- 7. Cierro el cursor

CLOSE c;

END;

/

**4. REF CURSOR en FUNCIÓN (dentro de un paquete)**

-- 1. Paquete

CREATE OR REPLACE PACKAGE pkg\_fn\_emp AS

FUNCTION get\_by\_dept(p\_deptno IN NUMBER) RETURN SYS\_REFCURSOR;

END pkg\_fn\_emp;

/

-- 2. Cuerpo

CREATE OR REPLACE PACKAGE BODY pkg\_fn\_emp AS

FUNCTION get\_by\_dept(p\_deptno IN NUMBER) RETURN SYS\_REFCURSOR IS

l\_cur SYS\_REFCURSOR; -- Cursor interno

BEGIN

-- 3. Apertura según filtro

OPEN l\_cur FOR

SELECT empno, ename, sal

FROM emp

WHERE deptno = p\_deptno;

RETURN l\_cur; -- 4. Devuelve cursor

END get\_by\_dept;

END pkg\_fn\_emp;

/

Uso:

DECLARE

c SYS\_REFCURSOR; -- 1. Cursor

e\_no emp.empno%TYPE; -- 2. empno

e\_nm emp.ename%TYPE; -- 3. ename

e\_sal emp.sal%TYPE; -- 4. sal

BEGIN

-- 5. Función del paquete devuelve cursor

c := pkg\_fn\_emp.get\_by\_dept(40);

-- 6. Bucle FETCH

LOOP

FETCH c INTO e\_no, e\_nm, e\_sal;

EXIT WHEN c%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(e\_no || '-' || e\_nm || '-' || e\_sal);

END LOOP;

-- 7. Cierre

CLOSE c;

END;

/

**5. BULK COLLECT INTO en PROCEDIMIENTO (sin paquete)**

-- 1. Procedimiento standalone

CREATE OR REPLACE PROCEDURE proc\_bulk\_emp\_by\_dept (

p\_deptno IN NUMBER, -- Número de dept.

p\_emps OUT emp\_tab -- OUT: colección emp\_tab

)

IS

BEGIN

-- 2. Bulk collect: llena la colección de golpe

SELECT emp\_obj(empno, ename, sal)

BULK COLLECT INTO p\_emps -- Variable OUT de tipo emp\_tab

FROM emp

WHERE deptno = p\_deptno;

END proc\_bulk\_emp\_by\_dept;

/

Uso:

DECLARE

v\_list emp\_tab; -- 1. Colección para recibir datos

BEGIN

-- 2. Invocación del procedimiento

proc\_bulk\_emp\_by\_dept(p\_deptno => 30,

p\_emps => v\_list);

-- 3. Iteración por índice

FOR i IN 1..v\_list.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

v\_list(i).empno || ' ' ||

v\_list(i).ename || ' ' ||

v\_list(i).sal

);

END LOOP;

END;

/

**6. BULK COLLECT INTO en PROCEDIMIENTO (dentro de paquete)**

-- 1. Paquete

CREATE OR REPLACE PACKAGE pkg\_bulk\_emp AS

PROCEDURE by\_dept(

p\_deptno IN NUMBER,

p\_emps OUT emp\_tab

);

END pkg\_bulk\_emp;

/

-- 2. Cuerpo

CREATE OR REPLACE PACKAGE BODY pkg\_bulk\_emp AS

PROCEDURE by\_dept(

p\_deptno IN NUMBER,

p\_emps OUT emp\_tab

)

IS

BEGIN

-- 3. Bulk collect en un paquete

SELECT emp\_obj(empno, ename, sal)

BULK COLLECT INTO p\_emps

FROM emp

WHERE deptno = p\_deptno;

END by\_dept;

END pkg\_bulk\_emp;

/

Uso:

DECLARE

v\_tab emp\_tab; -- 1. Colección

BEGIN

-- 2. Llamada

pkg\_bulk\_emp.by\_dept(20, v\_tab);

-- 3. Recorro colección

FOR i IN 1..v\_tab.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

v\_tab(i).empno || '-' ||

v\_tab(i).ename || '-' ||

v\_tab(i).sal

);

END LOOP;

END;

/

**7. BULK COLLECT INTO en FUNCIÓN (sin paquete)**

-- 1. Función standalone que devuelve colección

CREATE OR REPLACE FUNCTION fn\_bulk\_emp\_by\_dept (

p\_deptno IN NUMBER

) RETURN emp\_tab -- Retorna la colección emp\_tab

IS

l\_emps emp\_tab; -- Variable local

BEGIN

-- 2. Bulk collect en función

SELECT emp\_obj(empno, ename, sal)

BULK COLLECT INTO l\_emps

FROM emp

WHERE deptno = p\_deptno;

RETURN l\_emps; -- 3. Devuelve colección

END fn\_bulk\_emp\_by\_dept;

/

Uso:

DECLARE

l\_emps emp\_tab; -- 1. Recibe la colección

BEGIN

-- 2. Asignación desde función

l\_emps := fn\_bulk\_emp\_by\_dept(10);

-- 3. Iteración

FOR i IN 1..l\_emps.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

l\_emps(i).empno || ' ' ||

l\_emps(i).ename || ' ' ||

l\_emps(i).sal

);

END LOOP;

END;

/

**8. BULK COLLECT INTO en FUNCIÓN (dentro de paquete)**

-- 1. Paquete

CREATE OR REPLACE PACKAGE pkg\_fn\_bulk AS

FUNCTION by\_dept(p\_deptno IN NUMBER) RETURN emp\_tab;

END pkg\_fn\_bulk;

/

-- 2. Cuerpo

CREATE OR REPLACE PACKAGE BODY pkg\_fn\_bulk AS

FUNCTION by\_dept(p\_deptno IN NUMBER) RETURN emp\_tab IS

l\_emps emp\_tab; -- Variable local

BEGIN

-- 3. Bulk collect dentro del paquete

SELECT emp\_obj(empno, ename, sal)

BULK COLLECT INTO l\_emps

FROM emp

WHERE deptno = p\_deptno;

RETURN l\_emps; -- 4. Retorna la colección

END by\_dept;

END pkg\_fn\_bulk;

/

Uso:

DECLARE

v\_emps emp\_tab; -- 1. Colección

BEGIN

-- 2. Llamada a la función del paquete

v\_emps := pkg\_fn\_bulk.by\_dept(50);

-- 3. Muestra cada elemento

FOR i IN 1..v\_emps.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(

v\_emps(i).empno || '|' ||

v\_emps(i).ename || '|' ||

v\_emps(i).sal

);

END LOOP;

END;

/

Con estos ocho ejemplos tienes una **guía completa** para:

1. Devolver **CURSORS** (SYS\_REFCURSOR) desde procedimientos y funciones.
2. Recoger **colecciones** con BULK COLLECT INTO en procedimientos y funciones.
3. Implementarlo **sin paquetes** y **dentro de paquetes**, con comentarios línea a línea para que veas exactamente qué hace cada instrucción. ¡A practicar!