Aquí tienes algunas **"Reglas de Oro"** para trabajar con cursores en PL/SQL y evitar errores como ORA-01422: exact fetch returns more than requested number of rows:

**🔹 1. Usa SELECT INTO solo cuando estés seguro de que la consulta devuelve una sola fila**

✅ **Correcto (siempre devuelve una sola fila):**

DECLARE

V\_EMPLEADO EMP.ENAME%TYPE;

BEGIN

SELECT ENAME INTO V\_EMPLEADO FROM EMP WHERE EMPNO = 7369; -- Solo un resultado esperado

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || V\_EMPLEADO);

END;

/

❌ **Error si hay más de una fila:**

DECLARE

V\_EMPLEADO EMP.ENAME%TYPE;

BEGIN

SELECT ENAME INTO V\_EMPLEADO FROM EMP WHERE JOB = 'CLERK'; -- Puede devolver varias filas

END;

/

-- ERROR: ORA-01422: exact fetch returns more than requested number of rows

Si no estás seguro, usa **cursores explícitos**.

**🔹 2. Usa FETCH en un cursor explícito cuando esperas varias filas**

✅ **Correcto (procesa cada fila una por una con FETCH)**

DECLARE

CURSOR CUR\_EMPLEADOS IS

SELECT ENAME FROM EMP WHERE JOB = 'CLERK';

V\_AUX EMP.ENAME%TYPE;

BEGIN

OPEN CUR\_EMPLEADOS;

FETCH CUR\_EMPLEADOS INTO V\_AUX;

WHILE CUR\_EMPLEADOS%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || V\_AUX);

FETCH CUR\_EMPLEADOS INTO V\_AUX;

END LOOP;

CLOSE CUR\_EMPLEADOS;

END;

/

Esto evita errores porque **el FETCH va sacando las filas una a una**.

**🔹 3. Usa EXIT WHEN CURSOR%NOTFOUND en lugar de WHILE CURSOR%FOUND para mayor seguridad**

✅ **Correcto (termina correctamente al quedarse sin registros):**

DECLARE

CURSOR CUR\_EMPLEADOS IS

SELECT ENAME FROM EMP WHERE JOB = 'CLERK';

V\_AUX EMP.ENAME%TYPE;

BEGIN

OPEN CUR\_EMPLEADOS;

LOOP

FETCH CUR\_EMPLEADOS INTO V\_AUX;

EXIT WHEN CUR\_EMPLEADOS%NOTFOUND; -- La forma más segura

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || V\_AUX);

END LOOP;

CLOSE CUR\_EMPLEADOS;

END;

/

Esto es **mejor** que WHILE CURSOR%FOUND porque en WHILE la condición se evalúa antes del primer FETCH, lo que puede causar errores si el cursor no devuelve ninguna fila.

**🔹 4. Usa EXCEPTION WHEN NO\_DATA\_FOUND si SELECT INTO puede no encontrar datos**

✅ **Correcto (maneja el error si no hay resultados):**

DECLARE

V\_EMPLEADO EMP.ENAME%TYPE;

BEGIN

SELECT ENAME INTO V\_EMPLEADO FROM EMP WHERE EMPNO = 9999; -- Un número de empleado inexistente

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || V\_EMPLEADO);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No se encontró ningún empleado.');

END;

/

Si no usas EXCEPTION, el programa fallará si la consulta no encuentra datos.

**🔹 5. Cierra siempre los cursores abiertos**

✅ **Correcto (cierra el cursor para liberar memoria y evitar bloqueos):**

DECLARE

CURSOR CUR\_EMPLEADOS IS SELECT ENAME FROM EMP;

V\_AUX EMP.ENAME%TYPE;

BEGIN

OPEN CUR\_EMPLEADOS;

LOOP

FETCH CUR\_EMPLEADOS INTO V\_AUX;

EXIT WHEN CUR\_EMPLEADOS%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || V\_AUX);

END LOOP;

CLOSE CUR\_EMPLEADOS; -- ¡Siempre cerrar el cursor!

END;

/

Si olvidas cerrar el cursor, puede causar problemas de rendimiento y consumo de memoria.

**🔹 6. Usa FOR CURSOR si solo necesitas recorrer un cursor sin FETCH**

✅ **Correcto (simplifica el código evitando OPEN, FETCH y CLOSE)**

BEGIN

FOR EMP\_REC IN (SELECT ENAME FROM EMP WHERE JOB = 'CLERK') LOOP

DBMS\_OUTPUT.PUT\_LINE('Empleado: ' || EMP\_REC.ENAME);

END LOOP;

END;

/

💡 **Ventaja:**

* No necesitas OPEN, FETCH ni CLOSE.
* Más eficiente y fácil de leer.

**🔥 Resumen de las Reglas de Oro**

| **Regla** | **Explicación** |
| --- | --- |
| ✅ Usa SELECT INTO solo si hay **una fila garantizada** |  |
| ✅ Usa **cursores explícitos** si esperas más de una fila |  |
| ✅ Usa EXIT WHEN CURSOR%NOTFOUND en LOOP para evitar errores |  |
| ✅ Usa EXCEPTION WHEN NO\_DATA\_FOUND si el resultado puede estar vacío |  |
| ✅ **Cierra siempre los cursores** con CLOSE |  |
| ✅ Usa FOR CURSOR si solo necesitas recorrer datos sin FETCH |  |

Siguiendo estas reglas, evitarás errores y harás código PL/SQL más eficiente y seguro. 🚀

| **Uso** | **Mejor opción** |
| --- | --- |
| **Solo recorrer e imprimir datos** | FOR CURSOR ✅ |
| **Modificar datos, cálculos, llamadas a procedimientos** | WHILE con FETCH ✅ |
| **Control de errores personalizado** | WHILE con FETCH ✅ |

Si **solo imprimes datos**, usa FOR CURSOR.  
Si **haces más operaciones**, usa WHILE con FETCH. 🚀