| **Código** | **Nombre** | **Descripción** |
| --- | --- | --- |
| -00001 | DUP\_VAL\_ON\_INDEX | Se intenta insertar un valor duplicado en una columna con restricción UNIQUE o PRIMARY KEY. |
| -00942 | ORA-00942 | La tabla o vista no existe. |
| -01400 | ORA-01400 | Se intentó insertar NULL en una columna con restricción NOT NULL. |
| -01403 | NO\_DATA\_FOUND | No se encontraron filas en una consulta SELECT INTO. |
| -01422 | TOO\_MANY\_ROWS | Una consulta SELECT INTO devolvió más de una fila. |
| -01722 | ORA-01722 | Error de conversión, por ejemplo, intentar insertar un VARCHAR en un campo NUMBER. |
| -02291 | ORA-02291 | Se intenta insertar un valor en una clave foránea (FOREIGN KEY) que no existe en la tabla padre. |
| -02292 | ORA-02292 | Se intenta eliminar un registro con claves foráneas dependientes en otras tablas. |
| -04091 | ORA-04091 | Se detectó una mutación de tabla en un TRIGGER. |
| -06502 | VALUE\_ERROR | Error de desbordamiento numérico o incompatibilidad de tipo de datos. |
| -06504 | ROWTYPE\_MISMATCH | Tipos de datos incompatibles en una variable %ROWTYPE o %TYPE. |
| -06511 | CURSOR\_ALREADY\_OPEN | Se intentó abrir un cursor que ya estaba abierto. |
| -06512 | ORA-06512 | Error en una línea específica de un procedimiento o función almacenada. |
| -06519 | ORA-06519 | Se llamó a una función PL/SQL desde un entorno no compatible. |
| -06550 | ORA-06550 | Error de sintaxis en PL/SQL. |

**📌 CONTROL DE ERRORES EN PL/SQL: EXCEPCIONES**

**🔹 ¿Qué es una excepción?**

Una **excepción** en PL/SQL es un mecanismo que permite manejar errores que ocurren durante la ejecución del programa. Las excepciones pueden ser:

1. **Predefinidas** (maneja errores internos de Oracle automáticamente).
2. **Definidas por el usuario** (el programador las define y las lanza con RAISE).

📌 **Ventajas de usar excepciones:**  
✔️ Separa el código de control de errores del código principal.  
✔️ Mejora la legibilidad y mantenimiento del código.  
✔️ Permite manejar errores de manera controlada.

**🏗 Estructura de manejo de excepciones**

Cada bloque PL/SQL puede tener una sección EXCEPTION donde se capturan y manejan los errores.

**Sintaxis básica**

BEGIN

-- Código que puede generar errores

EXCEPTION

WHEN excepcion1 THEN

-- Código a ejecutar si ocurre excepcion1

WHEN excepcion2 OR excepcion3 THEN

-- Código a ejecutar si ocurre excepcion2 o excepcion3

WHEN OTHERS THEN

-- Código a ejecutar para cualquier otro error

END;

🔹 WHEN OTHERS es una cláusula especial que captura cualquier error que no haya sido manejado antes.  
🔹 Solo se puede usar **una vez** dentro de un bloque.

**🛠 Ejemplo básico de manejo de excepciones**

Supongamos que queremos consultar un vecino por apellido y manejar el error si no se encuentra:

DECLARE

v\_comunidad NUMBER(7);

BEGIN

BEGIN

-- Intentamos encontrar al vecino "SALVADOR"

SELECT pts\_comunidad INTO v\_comunidad

FROM vecinos

WHERE apellido1 LIKE 'SALVADOR';

EXCEPTION

-- Si no se encuentra, buscamos "CASTRILLO"

WHEN NO\_DATA\_FOUND THEN

SELECT pts\_comunidad INTO v\_comunidad

FROM vecinos

WHERE apellido1 LIKE 'CASTRILLO';

END;

-- Modificamos la comunidad si es menor a 7000

IF v\_comunidad < 7000 THEN

v\_comunidad := v\_comunidad + 1000;

END IF;

-- Insertamos el nuevo registro

INSERT INTO vecinos VALUES (8, 'A', 'ROBERTO', 'LOSADA', '22-JUN-99', v\_comunidad, v\_comunidad/166.386);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error inesperado: ' || SQLCODE || ' - ' || SQLERRM);

END;

/

✅ **Comentarios clave:**

* Se usa un BEGIN ... EXCEPTION ... END; interno para manejar NO\_DATA\_FOUND.
* WHEN OTHERS captura cualquier error no manejado y lo muestra en consola con SQLCODE y SQLERRM.

**🔥 Excepciones predefinidas en PL/SQL**

Oracle ya tiene excepciones predefinidas que puedes usar sin declararlas:

| **Excepción** | **Se produce cuando...** |
| --- | --- |
| CURSOR\_ALREADY\_OPEN | Se intenta abrir un cursor ya abierto. |
| DUP\_VAL\_ON\_INDEX | Se viola una restricción de unicidad en un índice. |
| INVALID\_CURSOR | Operación ilegal sobre un cursor. |
| INVALID\_NUMBER | Intento de convertir texto en número con caracteres inválidos. |
| NO\_DATA\_FOUND | Un SELECT INTO no devuelve filas. |
| TOO\_MANY\_ROWS | Un SELECT INTO devuelve más de una fila. |
| ZERO\_DIVIDE | Se intenta dividir por cero. |
| VALUE\_ERROR | Error de conversión o truncamiento de datos. |

**Ejemplo con ZERO\_DIVIDE**

DECLARE

v\_resultado NUMBER;

BEGIN

v\_resultado := 10 / 0; -- Esto genera un error de división por cero

EXCEPTION

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('Error: División por cero no permitida.');

END;

/

**🔹 Excepciones definidas por el usuario**

Si quieres manejar errores personalizados, puedes definir tus propias excepciones:

**Pasos para definir una excepción de usuario**

1. **Declararla en la sección DECLARE.**
2. **Activarla con RAISE dentro de BEGIN.**
3. **Manejarla en la sección EXCEPTION.**

**Ejemplo**

DECLARE

fecha\_nula EXCEPTION; -- Definir excepción

fecha\_no\_nula EXCEPTION;

CURSOR c\_vecinos IS SELECT \* FROM vecinos FOR UPDATE;

BEGIN

FOR i IN c\_vecinos LOOP

BEGIN

IF i.fecha\_llegada IS NULL THEN

RAISE fecha\_nula; -- Lanzar la excepción

END IF;

RAISE fecha\_no\_nula;

EXCEPTION

WHEN fecha\_nula THEN

UPDATE vecinos SET fecha\_llegada = SYSDATE WHERE CURRENT OF c\_vecinos;

WHEN fecha\_no\_nula THEN

UPDATE vecinos

SET pts\_comunidad = i.pts\_comunidad + 1000,

e\_comunidad = (i.pts\_comunidad + 1000) / 166.386

WHERE CURRENT OF c\_vecinos;

END;

END LOOP;

COMMIT;

END;

/

✅ **Explicación:**

* Si fecha\_llegada es NULL, se lanza la excepción fecha\_nula.
* WHEN fecha\_nula THEN actualiza fecha\_llegada.
* WHEN fecha\_no\_nula THEN actualiza pts\_comunidad.

**🚨 Errores internos con SQLCODE y SQLERRM**

Cuando ocurre un error de Oracle sin excepción predefinida, podemos obtener detalles con:

* SQLCODE → Código de error.
* SQLERRM → Mensaje de error.

**Ejemplo con WHEN OTHERS**

BEGIN

-- Código que puede fallar

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error ' || SQLCODE || ': ' || SQLERRM);

END;

/

**🚀 RAISE\_APPLICATION\_ERROR**

Para lanzar errores personalizados con mensaje propio, se usa:

RAISE\_APPLICATION\_ERROR(-20000, 'Mensaje de error');

📌 **Ejemplo práctico**

CREATE OR REPLACE PROCEDURE subir\_horas (emple NUMBER, horas\_subir NUMBER) IS

horas\_actuales NUMBER;

BEGIN

SELECT horas INTO horas\_actuales FROM empleados WHERE id\_empleado = emple;

IF horas\_actuales IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20010, 'El empleado no tiene horas registradas.');

ELSE

UPDATE empleados SET horas = horas\_actuales + horas\_subir WHERE id\_empleado = emple;

END IF;

END subir\_horas;

/

✅ **Notas:**

* RAISE\_APPLICATION\_ERROR(-20010, 'Mensaje') genera una excepción personalizada.
* Solo puede capturarse con WHEN OTHERS.

**🏆 📌 REGLAS DE ORO PARA EXCEPCIONES EN PL/SQL**

1️⃣ **Siempre usa una sección EXCEPTION en tus bloques PL/SQL.**  
2️⃣ **Usa WHEN OTHERS solo si no hay una excepción más específica.**  
3️⃣ **Nunca dejes errores sin manejar, usa DBMS\_OUTPUT.PUT\_LINE para depurar.**  
4️⃣ **Si defines excepciones personalizadas, siempre lánzalas con RAISE.**  
5️⃣ **Usa RAISE\_APPLICATION\_ERROR para mensajes de error personalizados.**  
6️⃣ **Recuerda que un error en la sección EXCEPTION finaliza el bloque.**  
7️⃣ **Las excepciones declaradas en DECLARE no pueden capturar errores en la inicialización.**

Con esto ya tienes una explicación clara y detallada. ¿Necesitas más ejemplos? 🚀

El error ORA-02291 indica que estás intentando insertar un valor en DETALLE\_FACTURAS.COD\_FACTURA que no existe en la tabla FACTURAS. Esto significa que la clave foránea (COD\_FACTURA) no tiene un valor correspondiente en la tabla padre.

**📌 ¿Cómo capturar ORA-02291 en PL/SQL?**

Usa EXCEPTION en el procedimiento para capturar SQLCODE = -2291.

**🔧 Código corregido con manejo de errores**

CREATE OR REPLACE PROCEDURE insertar\_detalle\_factura (

p\_cod\_factura IN DETALLE\_FACTURAS.COD\_FACTURA%TYPE,

p\_cod\_producto IN DETALLE\_FACTURAS.COD\_PRODUCTO%TYPE,

p\_cantidad IN DETALLE\_FACTURAS.CANTIDAD%TYPE

)

IS

CURSOR C\_SUBTOTAL IS

SELECT NVL(SUM(SUBTOTAL), 0)

FROM DETALLE\_FACTURAS

WHERE cod\_factura = p\_cod\_factura;

V\_AUX NUMBER;

V\_SQLCODE NUMBER;

V\_SQLERRM VARCHAR2(200);

BEGIN

OPEN C\_SUBTOTAL;

FETCH C\_SUBTOTAL INTO V\_AUX;

CLOSE C\_SUBTOTAL;

-- Intentar insertar en detalle\_facturas

INSERT INTO DETALLE\_FACTURAS (cod\_factura, cod\_producto, cantidad, subtotal)

VALUES (p\_cod\_factura, p\_cod\_producto, p\_cantidad, V\_AUX);

DBMS\_OUTPUT.PUT\_LINE('Detalle insertado correctamente con subtotal: ' || V\_AUX);

EXCEPTION

WHEN OTHERS THEN

V\_SQLCODE := SQLCODE;

V\_SQLERRM := SQLERRM;

-- Manejo específico de ORA-02291 (Clave foránea no encontrada)

IF V\_SQLCODE = -2291 THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: No existe la factura con COD\_FACTURA = ' || p\_cod\_factura);

DBMS\_OUTPUT.PUT\_LINE('Verifica que la factura exista antes de insertar el detalle.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('ERROR GENERAL:');

END IF;

DBMS\_OUTPUT.PUT\_LINE('SQLCODE: ' || V\_SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('SQLERRM: ' || V\_SQLERRM);

ROLLBACK; -- Deshacer la transacción si hay un error

END insertar\_detalle\_factura;

/

**🔄 Explicación paso a paso**

1. **Se captura el error con EXCEPTION WHEN OTHERS THEN**
   * Se usa SQLCODE y SQLERRM para obtener detalles del error.
2. **Se identifica si SQLCODE = -2291** (clave foránea no encontrada).
   * Si es ese error, se muestra un mensaje claro:  
     👉 "No existe la factura con COD\_FACTURA = X".
3. **Se aplica ROLLBACK** si ocurre un error.
   * Esto evita datos inconsistentes.

**🔍 ¿Cómo probarlo?**

Ejecuta este procedimiento con un COD\_FACTURA que **NO exista** en FACTURAS:

BEGIN

insertar\_detalle\_factura(9999, 101, 2); -- COD\_FACTURA 9999 no existe

END;

/

🔹 **Salida esperada:**

ERROR: No existe la factura con COD\_FACTURA = 9999

Verifica que la factura exista antes de insertar el detalle.

SQLCODE: -2291

SQLERRM: ORA-02291: restricción de integridad (VENTAS.SYS\_C0016226) violada - clave principal no encontrada

**✅ Conclusión**

✔ Ahora el procedimiento **detecta y maneja ORA-02291 correctamente**.  
✔ Evita insertar detalles en facturas inexistentes.  
✔ Da **mensajes claros** para que sepas qué corregir.

¿Necesitas otra validación o más detalles? 😊

No necesariamente. Puedes manejar errores específicos con WHEN, pero si quieres capturar cualquier otro error desconocido, usas WHEN OTHERS.

**📌 Dos formas de manejar errores en PL/SQL**

**1️⃣ Manejo específico de errores con WHEN**

Si ya sabes qué errores pueden ocurrir, puedes capturarlos directamente sin usar WHEN OTHERS.  
Ejemplo:

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: No se encontró el producto.');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Ya existe un registro con esa clave.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR DESCONOCIDO: ' || SQLCODE || ' - ' || SQLERRM);

✅ **Ventaja:** Controlas errores específicos sin afectar otros casos.  
❌ **Desventaja:** Si ocurre otro error no manejado, el procedimiento fallará.

**2️⃣ Manejo general con WHEN OTHERS**

Si no sabes qué errores pueden ocurrir o quieres capturarlos todos, usas WHEN OTHERS.  
Ejemplo:

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR DESCONOCIDO: ' || SQLCODE || ' - ' || SQLERRM);

✅ **Ventaja:** Captura **cualquier error** inesperado.  
❌ **Desventaja:** No distingue entre diferentes errores.

**3️⃣ Combinación de ambos (Recomendada)**

Usa WHEN para errores específicos y WHEN OTHERS para errores generales:

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: No se encontró el producto.');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Clave duplicada.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR GENERAL: ' || SQLCODE || ' - ' || SQLERRM);

✅ **Ventaja:**  
✔ Captura errores específicos con mensajes detallados.  
✔ Captura otros errores desconocidos con WHEN OTHERS.

**🔍 ¿Qué hacer en tu caso (ORA-02291)?**

Si solo quieres capturar ese error, usa esto:

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE = -2291 THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: No existe la factura con COD\_FACTURA = ' || p\_cod\_factura);

ELSE

DBMS\_OUTPUT.PUT\_LINE('ERROR DESCONOCIDO: ' || SQLCODE || ' - ' || SQLERRM);

END IF;

**✅ Conclusión**

📌 **¿Cuándo usar WHEN?**  
👉 Si quieres capturar errores específicos con un mensaje claro.

📌 **¿Cuándo usar WHEN OTHERS?**  
👉 Si quieres capturar cualquier error desconocido.

📌 **¿Mejor opción?**  
👉 **Usar ambos**: Capturar errores comunes con WHEN y manejar los desconocidos con WHEN OTHERS.

💡 **Así tu código será más robusto y fácil de depurar!** 😊

Sí, puedes obtener los códigos de error de varias maneras. Aquí te explico cómo hacerlo y te dejo una tabla con los errores más comunes en PL/SQL.

**📌 ¿Dónde encontrar los códigos de error en Oracle?**

1. **Desde la documentación oficial de Oracle**  
   👉 [Lista de errores de Oracle](https://docs.oracle.com/en/database/oracle/oracle-database/)
2. **Obtener el código del error en tiempo de ejecución**  
   Si tienes un error en tu procedimiento, usa:
3. BEGIN
4. -- Código que genera error
5. EXCEPTION
6. WHEN OTHERS THEN
7. DBMS\_OUTPUT.PUT\_LINE('SQLCODE: ' || SQLCODE);
8. DBMS\_OUTPUT.PUT\_LINE('SQLERRM: ' || SQLERRM);
9. END;
10. /

Esto imprimirá el código y el mensaje del error.

**📜 Tabla de errores más comunes en PL/SQL**

| **Código** | **Nombre** | **Descripción** |
| --- | --- | --- |
| -00001 | DUP\_VAL\_ON\_INDEX | Se intenta insertar un valor duplicado en una columna con restricción UNIQUE o PRIMARY KEY. |
| -00942 | ORA-00942 | La tabla o vista no existe. |
| -01400 | ORA-01400 | Se intentó insertar NULL en una columna con restricción NOT NULL. |
| -01403 | NO\_DATA\_FOUND | No se encontraron filas en una consulta SELECT INTO. |
| -01422 | TOO\_MANY\_ROWS | Una consulta SELECT INTO devolvió más de una fila. |
| -01722 | ORA-01722 | Error de conversión, por ejemplo, intentar insertar un VARCHAR en un campo NUMBER. |
| -02291 | ORA-02291 | Se intenta insertar un valor en una clave foránea (FOREIGN KEY) que no existe en la tabla padre. |
| -02292 | ORA-02292 | Se intenta eliminar un registro con claves foráneas dependientes en otras tablas. |
| -04091 | ORA-04091 | Se detectó una mutación de tabla en un TRIGGER. |
| -06502 | VALUE\_ERROR | Error de desbordamiento numérico o incompatibilidad de tipo de datos. |
| -06504 | ROWTYPE\_MISMATCH | Tipos de datos incompatibles en una variable %ROWTYPE o %TYPE. |
| -06511 | CURSOR\_ALREADY\_OPEN | Se intentó abrir un cursor que ya estaba abierto. |
| -06512 | ORA-06512 | Error en una línea específica de un procedimiento o función almacenada. |
| -06519 | ORA-06519 | Se llamó a una función PL/SQL desde un entorno no compatible. |
| -06550 | ORA-06550 | Error de sintaxis en PL/SQL. |

**📌 ¿Cómo usar estos códigos en tu EXCEPTION?**

Ejemplo de manejo de errores específicos en PL/SQL:

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Ya existe un registro con ese valor.');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: No se encontraron datos.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Se esperaba una fila, pero se devolvieron varias.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR DESCONOCIDO: ' || SQLCODE || ' - ' || SQLERRM);

**✅ Conclusión**

1️⃣ Usa SQLCODE y SQLERRM para ver los errores en tiempo real.  
2️⃣ Consulta la tabla de errores más comunes.  
3️⃣ Maneja errores específicos con WHEN.  
4️⃣ Usa WHEN OTHERS como respaldo para errores desconocidos.

Ahora tienes una guía rápida para identificar y manejar errores en PL/SQL. 🚀 ¿Te ayudo con algún error en particular? 😊

¡Exactamente! 🎯 Para los errores que **NO tienen un nombre predefinido**, usas WHEN OTHERS y dentro haces un IF SQLCODE = ... para capturar el código específico y mostrar un mensaje adecuado.

**📌 Resumen rápido**

✅ **Errores con nombres predefinidos** → Usas WHEN <NOMBRE\_DEL\_ERROR> THEN  
✅ **Errores sin nombre predefinido** → Usas WHEN OTHERS + IF SQLCODE = ... THEN

**🔹 Ejemplo con errores con nombre predefinido**

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: No se encontraron datos.');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR: Clave duplicada.');

🔹 Aquí NO\_DATA\_FOUND y DUP\_VAL\_ON\_INDEX ya tienen nombres predefinidos, por lo que puedes usarlos directamente.

**🔹 Ejemplo con errores sin nombre predefinido (como ORA-01400)**

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE = -1400 THEN -- ORA-01400: No se puede insertar NULL

DBMS\_OUTPUT.PUT\_LINE('ERROR: No se puede insertar NULL en una columna obligatoria.');

ELSIF SQLCODE = -2291 THEN -- ORA-02291: Violación de clave foránea

DBMS\_OUTPUT.PUT\_LINE('ERROR: Clave foránea no encontrada.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('ERROR GENERAL: ' || SQLCODE || ' - ' || SQLERRM);

END IF;

🔹 **Explicación**  
1️⃣ WHEN OTHERS captura cualquier error.  
2️⃣ IF SQLCODE = -1400 THEN maneja el error ORA-01400 (insertar NULL en columna NOT NULL).  
3️⃣ ELSIF SQLCODE = -2291 THEN maneja ORA-02291 (clave foránea no encontrada).  
4️⃣ ELSE captura cualquier otro error desconocido y lo imprime con SQLERRM.

**✅ Conclusión**

🔹 **Si el error tiene un nombre predefinido** → Usas WHEN <NOMBRE\_DEL\_ERROR> THEN.  
🔹 **Si el error NO tiene un nombre predefinido** → Usas WHEN OTHERS + IF SQLCODE = ... THEN.  
🔹 **Siempre deja un ELSE dentro de WHEN OTHERS** para manejar cualquier otro error.

Con esto, tu código será más robusto y fácil de depurar. 🚀 ¡Buena práctica! 💪 ¿Te ayudo con algún otro caso? 😊