

PL/SQL

Francisco Moreno

Universidad Nacional

Manejo de errores: *Exceptions*

- Una *exception* (excepción) surge cuando hay un error en tiempo de ejecución
- Cuando surge una excepción el proceso “salta” a la sección de manejo de las mismas (si no la hay se sale del (sub)bloque donde ocurrió): las instrucciones de esta sección se ejecutan y una vez finalizadas, el (sub)bloque se da por terminado

Hay dos tipos de excepciones:

1. **Excepciones predefinidas por Oracle**

- Son un conjunto de errores definidos en Oracle. No hay que declararlas.
- Son disparadas **automáticamente** por PL/SQL como respuesta a un error

2. **Excepciones definidas por el usuario**

- Se deben declarar
- Se deben disparar **explícitamente**

Algunas de las excepciones predefinidas:

- `TOO_MANY_ROWS`
- `INVALID_CURSOR`
(Ej: Cerrar un cursor que ya estaba cerrado)
- `NO_DATA_FOUND`
- `CURSOR_ALREADY_OPEN`
(Ej: Abrir un cursor que ya estaba abierto)
- `INVALID_NUMBER`
(Ej: Fallo de conversión*,
 'k3b' no es un número)
- `VALUE_ERROR`
(Ej: Error de truncamiento o conversión en una vble.)
- `ZERO_DIVIDE`
- `DUP_VAL_ON_INDEX`

* Dentro de una sentencia SQL

Sea la tabla:

```
DROP TABLE emp;
```

```
CREATE TABLE emp(  
    cod NUMBER(8) PRIMARY KEY,  
    nom VARCHAR2(15),  
    sueldo NUMBER(3)  
);
```

```
INSERT INTO emp VALUES (12, 'Kylie', 10);
```

```
INSERT INTO emp VALUES (15, 'Mavie', 5);
```

```
INSERT INTO emp VALUES (76, 'Diamanda', 15);
```

DECLARE

nro_emp emp.cod%TYPE;

BEGIN

SELECT cod INTO nro_emp

FROM emp

WHERE nom = 'Kylie';

DBMS_OUTPUT.PUT_LINE('El código de Kylie es: ' ||
nro_emp);

EXCEPTION

WHEN **NO_DATA_FOUND** THEN

DBMS_OUTPUT.PUT_LINE('Kylie no existe');

WHEN **TOO_MANY_ROWS** THEN

DBMS_OUTPUT.PUT_LINE('Hay varias Kylies');

END;

/

```
CREATE TABLE apuesta(cod NUMBER(8) PRIMARY KEY,  
                      cant NUMBER(8) NOT NULL);
```

```
DECLARE
```

```
cod_apta apuesta.cod%TYPE;
```

```
nro_ale apuesta.cant%TYPE;
```

```
BEGIN
```

S
u
b
l
o
q
u
e

```
FOR i IN 1..20 LOOP
```

```
  BEGIN
```

```
    cod_apta := ABS(MOD(DBMS_RANDOM.RANDOM, 10));
```

```
    nro_ale := ABS(MOD(DBMS_RANDOM.RANDOM, 20));
```

```
    INSERT INTO apuesta VALUES(cod_apta, nro_ale);
```

```
    EXCEPTION
```

```
    WHEN DUP_VAL_ON_INDEX THEN
```

```
      UPDATE apuesta SET cant = cant + nro_ale
```

```
      WHERE cod = cod_apta;
```

```
  END;
```

```
END LOOP;
```

```
END;
```

```
/
```

Cuando el usuario define sus excepciones, se procede así:

- Se declaran por medio del **tipo de datos**

EXCEPTION:

`nombre_excepcion EXCEPTION;`

- Se deben “disparar” mediante la sentencia

`RAISE nombre_excepcion;`


```
DECLARE
  nomina NUMBER;
  muy_alta EXCEPTION;
BEGIN
  SELECT SUM(sueldo) INTO nomina
  FROM emp;
  IF nomina > 100 THEN
    RAISE muy_alta;
  END IF;
  DBMS_OUTPUT.PUT_LINE('El valor es: ' || nomina);
EXCEPTION
  WHEN muy_alta THEN
    DBMS_OUTPUT.PUT_LINE('Nomina muy alta');
END;
/
```

Nota: COUNT, MAX, MIN, SUM, AVG nunca generan las excepciones NO_DATA_FOUND ni TOO_MANY_ROWS, ni siquiera si la tabla está vacía ¿por qué?

El manejo de OTHERS


- Es una excepción predefinida que sirve para capturar una excepción que no ha sido tratada en el manejador de excepciones
- Se puede obtener el mensaje del error que ocurrió mediante la función `SQLERRM`
- Para el control de excepciones, como mínimo se debería usar `OTHERS` en todo programa PL/SQL
- `OTHERS` se debe escribir **de última** en el manejador de excepciones

Nota. El siguiente ejemplo requiere el permiso:

```
GRANT EXECUTE ON SYS.DBMS_LOCK TO usuario;
```

En una sesión ejecutar esta:

```
BEGIN
UPDATE emp
SET sueldo = sueldo + 1;
DBMS_LOCK.SLEEP(30);
COMMIT;
END;
/
```

Y a continuación en **otra** sesión ejecutar esta: 

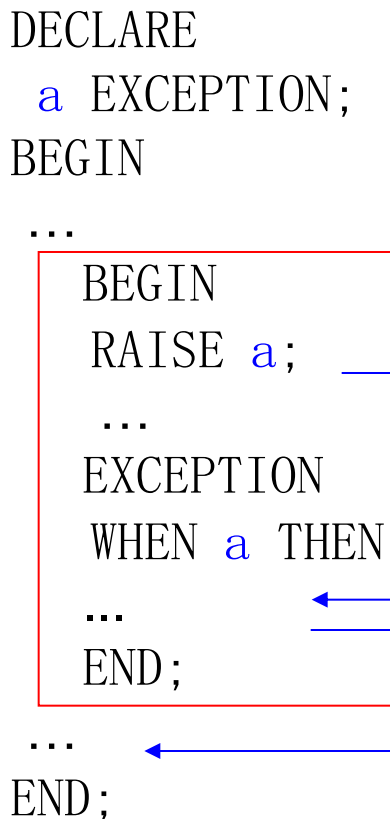
Ensayar sin el NOWAIT
y ver la diferencia

```
DECLARE
CURSOR emp_c IS SELECT * FROM emp
FOR UPDATE NOWAIT;
BEGIN
LOOP
BEGIN
OPEN emp_c; --Intenta bloquear
DBMS_OUTPUT.PUT_LINE('¡OK!');
EXIT; --Sale del ciclo
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Tabla ocupada
' || SQLERRM);
DBMS_LOCK.SLEEP(5); --Esperar 5 seg
END;
END LOOP;
COMMIT;
END;
/
```

Propagación de las excepciones

Caso 1:

```
DECLARE
  a EXCEPTION;
BEGIN
  ...
  BEGIN
    RAISE a;
    ...
    EXCEPTION
    WHEN a THEN
      ...
    END;
  ...
END;
```



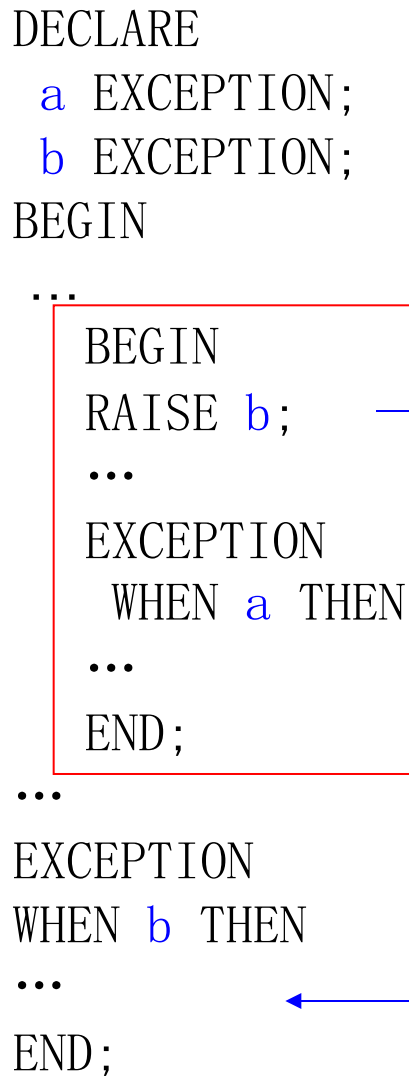
-- Se dispara la excepción a

-- Se trata la excepción aquí

-- El control continúa aquí

Caso 2:

```
DECLARE
  a EXCEPTION;
  b EXCEPTION;
BEGIN
  ...
  BEGIN
    RAISE b;
    ...
    EXCEPTION
      WHEN a THEN
        ...
      END;
  ...
  EXCEPTION
    WHEN b THEN
      ...
  END;
```

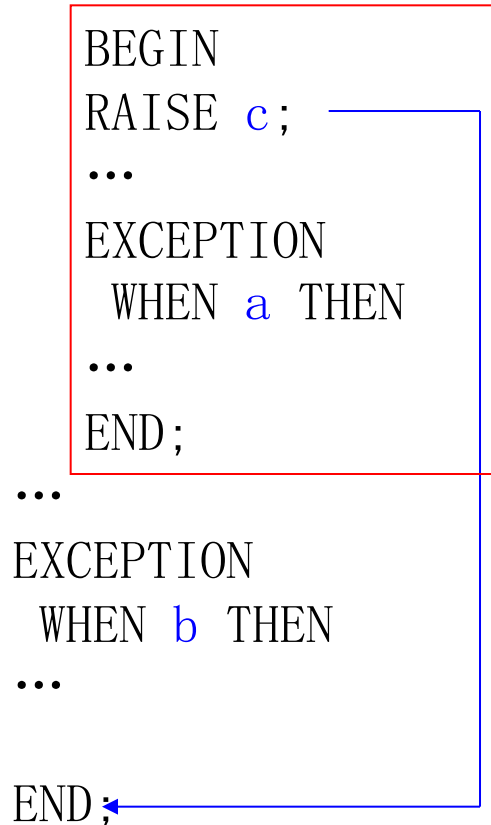


-- No se trató b aquí

-- El control continúa aquí

Caso 3:

```
DECLARE
  a EXCEPTION; b EXCEPTION;
  c EXCEPTION;
BEGIN
  ...
  BEGIN
    RAISE c;
    ...
    EXCEPTION
      WHEN a THEN
        ...
  END;
  ...
EXCEPTION
  WHEN b THEN
    ...
END;
```



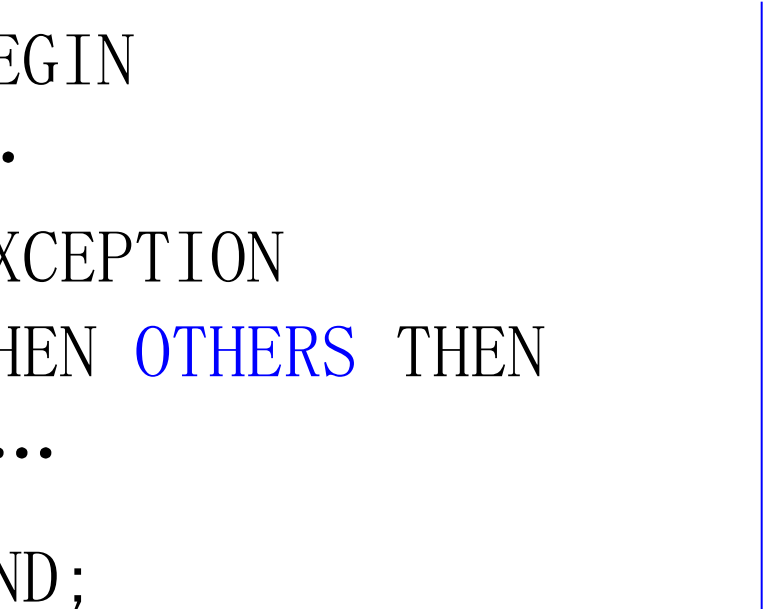
-- No se trató c aquí

-- No se trató tampoco c aquí

-- Aborta

Caso 4: Error en la zona de declaración

```
DECLARE
val NUMBER(5) := 'hola';
BEGIN
...
EXCEPTION
WHEN OTHERS THEN
...
END;
```

A blue line starts from the word 'OTHERS' in the 'WHEN OTHERS THEN' clause, goes vertically up, then horizontally to the right, and finally vertically down to an arrow pointing at the '-- Aborta' comment.

-- No la captura

-- Aborta

Caso 5: Error en la zona de declaración de un sub-bloque

```
BEGIN
```

```
...
```

```
  DECLARE
```

```
  x NUMBER(5) := 'Hola';
```

```
  BEGIN
```

```
    ...
```

```
  EXCEPTION
```

```
  WHEN OTHERS THEN
```

```
    ...
```

```
  END;
```

```
...
```

```
EXCEPTION
```

```
WHEN OTHERS THEN
```

```
... ←
```

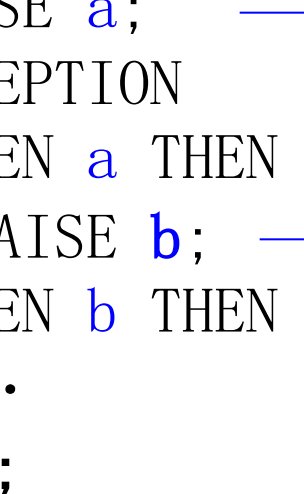
```
END;
```

-- No la captura

-- Se trata aquí

Caso 6: Excepciones disparadas dentro del bloque EXCEPTION

```
DECLARE
  a EXCEPTION;
  b EXCEPTION;
BEGIN
  ...
  RAISE a;
  EXCEPTION
    WHEN a THEN
      RAISE b;
    WHEN b THEN
      ...
  END;
```



-- a se trata aquí

-- No captura a b

-- Aborta

Caso 7: Excepciones disparadas dentro del bloque Exception

```
DECLARE
  a EXCEPTION; b EXCEPTION;
BEGIN
  ...
  BEGIN
    RAISE a;
    EXCEPTION
    WHEN a THEN
      RAISE b;
    WHEN b THEN
      ...
    END;
  EXCEPTION
  WHEN b THEN
    ...
  END;
```


-- a se trata aquí

-- No captura a b

-- b se trata aquí

Caso 8: OTHERS captura una excepción de usuario no tratada:

```
DECLARE
  a EXCEPTION;
BEGIN
  ...
  RAISE a;
  ...
EXCEPTION
  WHEN OTHERS THEN
    ...
END;
```

A blue line starts from the 'a' in 'RAISE a;' and extends to the right. It then turns downwards and then left, ending with an arrowhead pointing to the 'OTHERS' in the 'WHEN OTHERS THEN' clause of the exception handler.

-- a se captura aquí