

PL SQL error handling

exception propagates outward to each successive block, until properly handled or presented to client finally

EXCEPTION without it, error present to client

WHEN exception_name trap an exception

THEN ... handle an exception

RAISE raise the current exception

Types of Exception

Named System Exceptions

Unnamed System Exceptions

User-defined Exceptions

Named System Exceptions

NO_DATA_FOUND SELECT...INTO with no returned

ZERO_DIVIDE

TOO_MANY_ROWS fetch >1 row into a record/var

INVALID_CURSOR cursor closed

Unnamed system exceptions

WHEN OTHERS OTHERS handler used
THEN

to handle exception_name
explicitly EXCEPTION;

PRAGMA EXCEPTION_INIT (exception_name, Err_code);

User defined exception

my_exception raise my_exception;
EXCEPTION;

pragma exception_init(my_exception, -20001); assign error code

RAISE_APPLICATION_ERROR(code, msg) assign a user friendly message

WHEN my_exception trap and handle
THEN

Explicitly declared in declaration, raised in execution section and handled in Exception section

raise_application_error(code, msg)

predefined oracle procedure for user raise error with code and message

1) error code range(-20,000 ~ -20,999 required)

2) message varchar2(2000)

3) boolean: default true: adds the error to the top of list
false-added to top of list of all errors
current stack

from DBMS_STANDARD, no name provided, handled with OTHERS handler.
Uncommitted session tx rollback

raise_application_error with handler

```
DECLARE
    n_salary NUMBER;
    sal_high EXCEPTION;
    pragma exception_init(sal_high, -20001);
BEGIN
    SELECT salary INTO n_salary
    FROM employees
    WHERE employee_id = :emp_id;

    IF n_salary > 10000 THEN
        raise_application_error(
            -20001, 'Salary is high');
    END IF;
EXCEPTION
    WHEN sal_high THEN
        dbms_output.put_line('SQL CODE);
        dbms_output.put_line('SQL error);
END;
```

raise; //raise current error