TECHNICAL REFERENCE

HANDLE PL/SQL ERRORS: OVERVIEW

The following excerpt is from *PL/SQL User's Guide and Reference Version 1.0*, 800-V1.0-0991, pages 6-25 through 6-29.

Exception Handlers

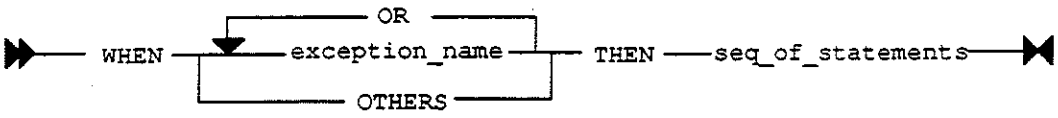
Description

An exception handler executes statements in response to a raised exception. When an exception is raised, normal execution of the PL/SQL block stops, and the statements in the appropriate exception handler are executed.

After the exception handler completes, processing of the block stops, and control returns to whatever invoked the block, such as an enclosing block or a precompiled program.

Syntax

```
exception_handler ::=
```



Keyword and Parameter Description

exception_name

This parameter is the name of a PL/SQL predefined exception or the name of a user-defined exception. Following is a list of the predefined exceptions:

CURSOR_ALREADY_OPEN

SQLCODE returns -6511 (ORACLE error code ORA-06511). Raised when you try to OPEN an already open cursor. You must CLOSE a cursor before you can reOPEN it.

A cursor FOR loop automatically opens the cursor it names. Therefore, you cannot enter the loop if that cursor is already open. Nor can you OPEN that cursor inside the loop.

DUP_VAL_ON_INDEX

SQLCODE returns -1 (ORACLE error code ORA-00001). Raised when an INSERT or UPDATE attempts to create two rows with duplicate values in a column constrained by a UNIQUE index.

INVALID_CURSOR

SQLCODE returns -1001 (ORACLE error code ORA-01001). Raised when a PL/SQL call specifies an invalid cursor (closing an unopened cursor, for example).

Handle PL/SQL Errors: Overview—cont'd

INVALID_NUMBER

SQLCODE returns -1722 (ORACLE error code ORA-01722). Raised in a SQL statement when the conversion of a character string to a number fails because the string does not represent a valid number. For example, the following INSERT statement raises INVALID_NUMBER when ORACLE tries to convert 'GREEN' to a number:

```
INSERT INTO emp (empno, ename, deptno) VALUES ('GREEN', 7888, 20);
```

In non-SQL statements, that is, in PL/SQL statements that contain no SQL commands, VALUE_ERROR is raised instead of INVALID_NUMBER. For example, the following assignment statement raises VALUE_ERROR, not INVALID_NUMBER:

```
DECLARE
    my_empno  NUMBER(4);
    my_ename  CHAR(10);
BEGIN
    my_empno := 'GREEN';
     ...
```

LOGIN_DENIED

SQLCODE returns -1017 (ORACLE error code ORA-01017). Raised when an invalid username/password was used to log on to ORACLE.

NO_DATA_FOUND

SQLCODE returns +100 (ORACLE error code ORA-01403). Raised when a single-row SELECT returns no rows. NO_DATA_FOUND is also raised by a DELETE or UPDATE that uses CURRENT OF cursor_name if the named cursor is open, but no FETCH has been done or the last FETCH returned no rows.

SQL group functions such as AVG, MAX, and SUM always return a value, even if the value is NULL. So, a single-row SELECT that calls a group function will never raise NO_DATA_FOUND because some data is always "found."

Because NO_DATA_FOUND is raised if a single-row SELECT returns no rows, you can check the value of SQL%NOTFOUND only in an exception handler. However, you can check the value of cursor_name%NOTFOUND after every FETCH. FETCH is expected to return no rows eventually, so when that happens, no exception is raised.

NOT_LOGGED_ON

SQLCODE returns -1012 (ORACLE error code ORA-01012). Raised when PL/SQL issues an ORACLE call without being logged on to ORACLE.

Handle PL/SQL Errors: Overview—cont'd

PROGRAM_ERROR

SQLCODE returns -6501 (ORACLE error code ORA-06501). Raised when PL/SQL has an internal problem.

STORAGE_ERROR

SQLCODE returns -6500 (ORACLE error code ORA-06500). Raised when PL/SQL runs out of memory or when memory is corrupted.

TIMEOUT_ON_RESOURCE

SQLCODE returns -51 (ORACLE error code ORA-00051). Raised when a timeout occurs while ORACLE is waiting for a resource. This error is usually caused by an instance that has terminated abnormally.

TOO_MANY_ROWS

SQLCODE returns -1427 (ORACLE error code ORA-01427). Raised when a SELECT statement returns more than one row.

VALUE_ERROR

SQLCODE returns -6502 (ORACLE error code ORA-06502). Raised when a string, arithmetic, numeric, conversion, or constraint error occurswith one exception (see INVALID_NUMBER).

If a string value assigned to a database column using INSERT or UPDATE is truncated, or if a string value assigned to a PL/SQL variable is truncated, VALUE_ERROR is raised.

However, if a string value assigned to a host variable is truncated, no exception is raised (but, if an indicator variable is used, PL/SQL sets it to the string's pre-truncation length). Numeric overflow and underflow errors are handled in the usual way.

ZERO_DIVIDE

SQLCODE returns -1476 (ORACLE error code ORA-01476). Raised when you attempt to divide a number by zero.

ORACLE error messages are listed in the ORACLE RDBMS Error Messages and Codes Manual. PL/SQL error messages are listed in Appendix F.

You can have multiple exceptions execute the same set of statements by following the WHEN keyword with a list of the exceptions, separating them by the OR keyword. If any of the exceptions named in the list are raised, the associated statements are executed.

You can have many different exception handlers if you have many different WHEN clauses. Each one can associate a different set of statements with a set of exceptions. An exception name, however, must appear only once in the exception handling part of the block.

Handle PL/SQL Errors: Overview—cont'd

OTHERS

OTHERS stands for all the other exceptions not explicitly named in the exception handling part of the block.

The use of OTHERS is optional and is allowed only as the last exception handler. You cannot include OTHERS in a list of exceptions following the WHEN keyword.

seq_of_statements

This parameter signifies a sequence of statements. For a definition of seq_of_statements, see the section "Block Structure" in Chapter 5.

Usage Notes

The exception handlers for a block must come at the end of the block. They are introduced by the keyword EXCEPTION, which must appear once before the first use of the keyword WHEN. The exception handling part of the block is terminated by the same END keyword that terminates the entire block.

An exception should be raised only when an error occurs that makes it impossible, impractical, or undesirable to finish processing.

- If there is no exception handler in the current block for a raised exception, the exception propagates according to the following rules: If there is an enclosing block for the current block, the exception is passed on to that block. The enclosing block then becomes the current block. If a handler for the raised exception is not found, the process repeats.
- If there is no enclosing block for the current block, the exception is passed back to the environment that invoked the PL/SQL program, such as SQL*Plus or a precompiled program.

Only one exception at a time can be active in the exception handling part of a block. Therefore, if an exception is raised inside a handler, the block that encloses the current block is the first block searched to find a handler for the newly raised exception. From there on, the exception propagates normally.

An exception handler can reference only those variables that the current block can reference.

Handle PL/SQL Errors: Overview—cont'd

Example

```
DECLARE
     bad_employee_num    EXCEPTION;
     bad_act_num         EXCEPTION;
     ...          -- other declarations go here
BEGIN
      ...         -- PL/SQL statements go here
EXCEPTION
     WHEN bad_employee_num OR bad_acct_num THEN  -- user defined
        ROLLBACK;
     WHEN ZERO_DIVIDE THEN   -- a predefined exception
        INSERT INTO inventory_table VALUES (prod_name, quantity);
        COMMIT;
END;
```

NAME AN ORACLE ERROR

The following excerpt is from *PL/SQL User's Guide and Reference Version 1.0*, 800-V1.0-0991, pages 6-30 through 6-31.

EXCEPTION_INIT

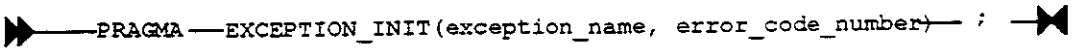
Description

EXCEPTION_INIT assigns a name to an ORACLE error code.

EXCEPTION_INIT lets you refer to internal exceptions by name rather than by using the OTHERS keyword in an exception handler.

Syntax

```
exception_init_pragma ::=
```



Keyword and Parameter Description

PRAGMA — This keyword signifies that the statement is a compiler directive, which is not processed when the PL/SQL block is executed. Such a statement is called a pragma.

exception_name — This parameter is the name of an exception previously declared using an exception_declaration.

error_code_number — The error_code_number can be any valid ORACLE error code. These are the same error codes that the function SQLCODE returns.

Usage Notes

The pragma must appear in the same declarative section as the associated exception, somewhere after the exception declaration. Thus, the EXCEPTION_INIT pragma can be used only in the declarative part of a PL/SQL block.

Be sure to assign only one exception name to an error code.

Name an ORACLE Error—cont'd

Example

```
DECLARE
    fetch_out_of_sequence       EXCEPTION;
    PRAGMA EXCEPTION_INIT(fetch_out_of_sequence, -1002);
         /* -1002 is the ORACLE error code generated when    *
         * a cursor was declared using the FOR UPDATE OF    *
         * clause, and a FETCH is attempted after a COMMIT  *
         * is performed.                                    */
BEGIN
    ...
    -- Put operations on cursors here.
    ...
EXCEPTION
    WHEN fetch_out_of_sequence THEN
         -- handle the error
    ...
END;
```

OPPGAVER FOR KURS 210 - PL/SQL

Dette oppgavesettet er tilleggsoppgaver til de fire første kapitlene i kursdokumentasjonen.

Dette dokumentet er utviklet i Frame Maker v.3.0.

OPPGAVER til KAP 2

1. Gi alle selgere som har provisjon på mer enn 250 i tabellen emp, en lønnsøkning på 10 %. Rull tilbake oppdateringen dersom mer enn 2 selgere justeres.

2. Legg inn i tabellen temp: empno, forslag til ny lønn og ename. Forslag til ny lønn skal være: gammel lønn + 7 % av nåværende lønn og provisjon.

OPPGAVER til KAP3 - Del 1

1. Lag et program som for gitt ansattnummer øker lønnen ut fra følgende satser:

JOB	% RAISE
ANALYST	6
CLERK	5.5
MANAGER	9
SALESMAN	3
PRESIDENT	0

OPPGAVER til KAP3 - DEL 2

1. Ta imot en tekststreng fra bruker, og legg inn den inverterte tekststrengen(tekststrengen baklengs) i temp tabellen.

2. Gi alle ansatte 5 % lønnsøknig. Gjenta justeringen helt til samlet lønnsvolum er så nær 50.000 som mulig.

OPPGAVER til KAP4

1. Legg inn de 10 høyeste lønningene i temp tabellen. Slik at den med høyest lønn kommer først. Videre skal de ansatte rangeres slik at de 4 første radene vil se ut som følger:

SAL	RANG	ENAME
5000	1	KING
3000	2	SCOTT
3000	2	FORD
2975	4	JONES

2. Alle ansatte skal ha en medarbeidersamtale 180 dager etter ansettelse. Gå igjennom alle ansatte og legg inn Hiredate, dato for samtale og Ename i temp tabellen. Sørg for at dato for medarbeidersamtale ikke blir lørdag eller søndag. I slike tilfelle velges fredagen før.

FASIT FOR OPPGAVER TIL KURS 210 - PL/SQL

Dette oppgavesettet er tilleggsoppgaver til de fire første kapitlene i kursdokumentasjonen.

Dette dokumentet er utviklet i Frame Maker v.3.0.

OPPGAVER til KAP 2

OPPGAVE 1:

```
BEGIN
update emp
   set sal = sal*1.1
where job='SALESMAN' and COMM > 250;

if SQL%rowcount > 3 then
  rollback;
else
  commit;
end if;
END;
/
```

OPPGAVE 2:

```
BEGIN
INSERT into temp
   select empno,
          sal + 0.07*(sal + nvl(comm,0)),
          ename
   from emp;
END;
/
```

OPPGAVER til KAP3 - Del 1

OPPGAVE 1:

```
DECLARE v_empno     emp.empno%TYPE := &p_empno;
        v_job       emp.job%TYPE;
BEGIN
    select job into v_job
    from emp
    where empno = v_empno;
if v_job = 'ANALYST' then
   update emp
      set sal= sal*1.06
   where empno=v_empno;
elsif v_job= 'CLERK' then
   update emp
      set sal= sal*1.055
   where empno=v_empno;
elsif v_job= 'MANAGER' then
  update emp
      set sal= sal*1.09
   where empno=v_empno;
elsif v_job = 'SALESMAN' then
  update emp
      set sal= sal*1.03
   where empno=v_empno;
end if;
END;
/
```

OPPGAVER til KAP3 - DEL 2

OPPGAVE 1:

Set serveroutput on

```
DECLARE V_tekst   varchar2(60)  := '&p_tekst';
        V_invers  varchar2(60);
BEGIN
for i in reverse 1..length(v_tekst) loop
   v_invers := v_invers || substr(v_tekst,i,1);
end loop;
dbms_output.put_line(v_invers);
END;
/
```

OPPGAVE 2:

```
DECLARE v_lonn_vol  number;
BEGIN
    select sum(sal) into v_lonn_vol
    from emp;
LOOP
    v_lonn_vol := v_lonn_vol *1.05;
    exit when v_lonn_vol > 50000;
    update emp set sal = sal*1.05;
end loop;
END;
/
```

OPPGAVER til KAP4

OPPGAVE 1:

```
DECLARE  v_rang number;
              v_gml_sal     emp.sal%TYPE := 0;
cursor c_emp is
               Select sal, ename
               from emp
               order by sal desc;

BEGIN
    for r_emp in c_emp loop
        if v_gml_sal != r_emp.sal then
           v_rang := C_emp%rowcount;
        end if;
        v_gml_sal := r_emp.sal;
        insert into temp
               values(r_emp.sal,
                      v_rang,
                      r_emp.ename);
        exit when C_emp%rowcount = 10;
    end loop;
END;
/
```

OPPGAVE 2:

```
DECLARE cursor c_emp is select hiredate,
                               hiredate+180 dato,
                               ename,
                               to_char(hiredate+180,'DY') dag
                          from emp;
        v_dato   date;

BEGIN

FOR r_emp in c_emp LOOP
    if r_emp.dag = 'SAT' then
       v_dato := r_emp.dato-1;
    elsif r_emp.dag = 'SUN' then
       v_dato := r_emp.dato-2;
    else
       v_dato := r_emp.dato;
    end if;
    insert into temp values(0,0,
                          to_char(r_emp.hiredate,'DD-MON-YY-DY')||' '||
                          to_char(v_dato,'DD-MON-YY-DY')||' '||
                          r_emp.ename);
END LOOP;
END;
/
```