

PL/SQL

Exceptions



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be



Inleiding



- **Compilatiefouten**

➤ Zoeken via “show errors”

- **Runtime-fouten**

= exception

van belang om bij de ontwikkeling v.e. applicatie na te gaan welke exceptions kunnen voorkomen



opvangen

het optreden van een exception

= het raisen van een exception

Inleiding: voorbeeld

```
CREATE OR REPLACE PROCEDURE get_naam
(p_job_id employees.job_id%type)
AS
    v_naam employees.last_name%type;
BEGIN
    SELECT last_name
    INTO v_naam
    FROM employees
    WHERE job_id = p_job_id;
    DBMS_OUTPUT.PUT_LINE(v_naam);
END;
```

- | | | |
|------------------------------------|--------------------------|-------------------------------|
| ➤ 1 persoon met de betreffende job | geen probleem | } exception → prog afgebroken |
| ➤ Geen persoon | melding: no data found | |
| ➤ Meerdere personen | melding: exact fetch ... | |

Inleiding: zelfde voorbeeld met exceptions

```
CREATE OR REPLACE PROCEDURE get_naam
(p_job_id employees.job_id%type)
AS
    v_naam          employees.last_name%type;
BEGIN
    SELECT last_name
    INTO v_naam
    FROM employees
    WHERE job_id = p_job_id;
    DBMS_OUTPUT.PUT_LINE(v_naam);
EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('Er is niemand met deze job');
    WHEN too_many_rows THEN
        DBMS_OUTPUT.PUT_LINE('Meer dan 1 persoon gevonden');
END;
```

Example of an Exception

```
DECLARE
  v_lname VARCHAR2(15);
BEGIN
  SELECT last_name INTO v_lname
  FROM employees
  WHERE first_name='John';
  DBMS_OUTPUT.PUT_LINE ('John's last name is :'
                        || v_lname);
END;
```

Error report:

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

ORA-06512: at line 4

01422. 00000 - "exact fetch returns more than requested number of rows"

*Cause: The number specified in exact fetch is less than the rows returned.

*Action: Rewrite the query or change number of rows requested

Example of an Exception

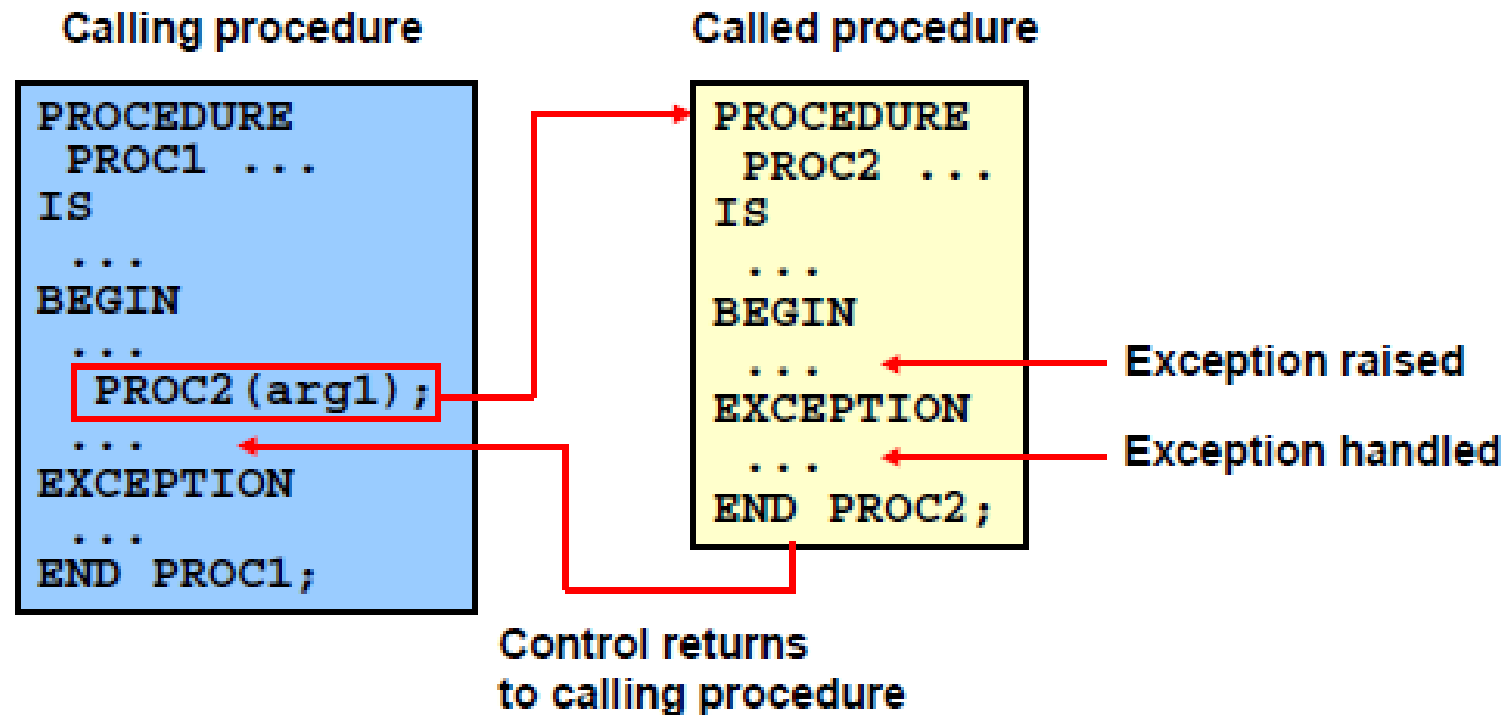
```
DECLARE
    v_lname VARCHAR2(15);
BEGIN
    SELECT last_name INTO v_lname
    FROM employees
    WHERE first_name='John';
    DBMS_OUTPUT.PUT_LINE ('John''s last name is : '
                          || v_lname);

EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE (' Your select statement
        retrieved multiple rows. Consider using a
        cursor. ');

END;
/
```

```
anonymous block completed
Your select statement retrieved multiple
rows. Consider using a cursor.
```

Handled Exceptions



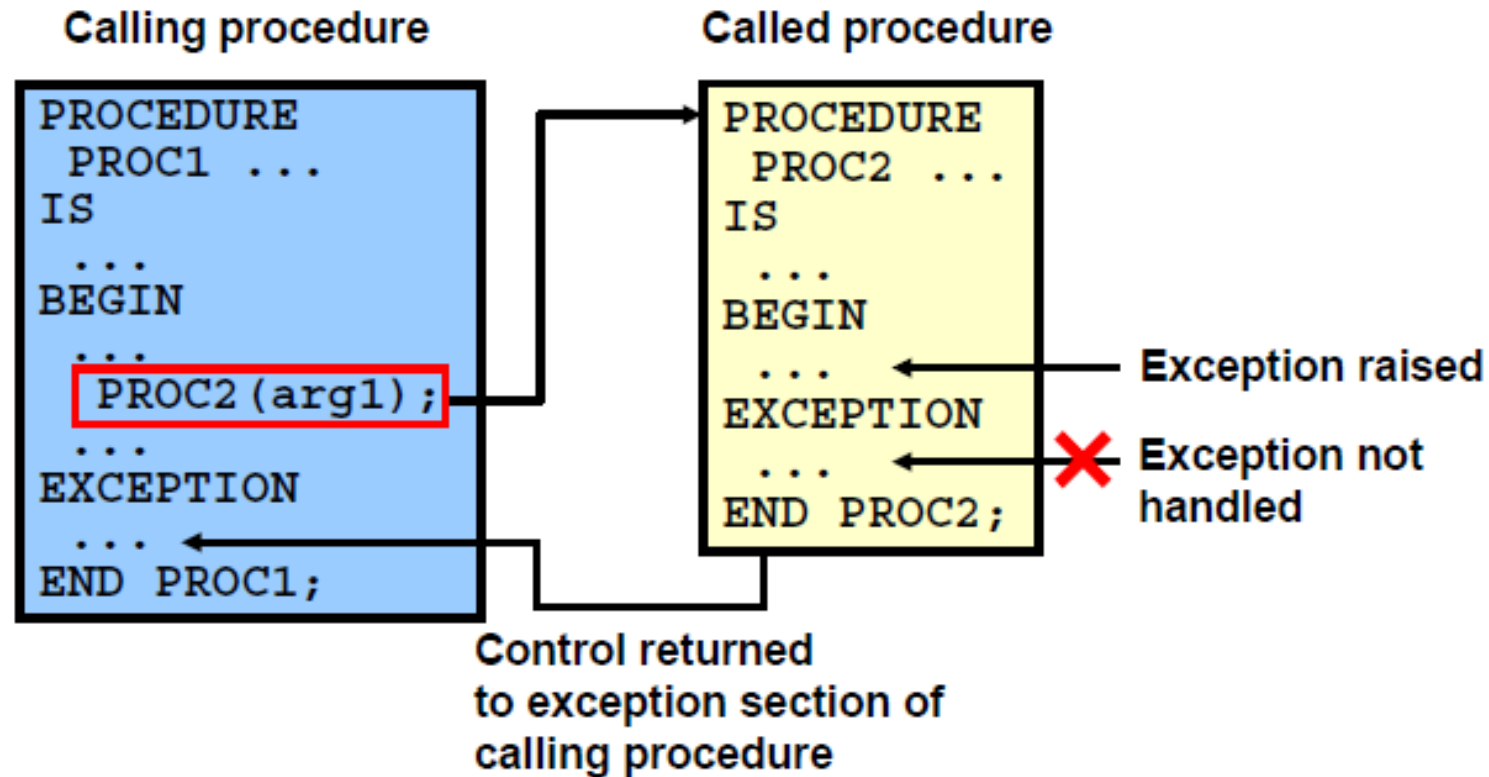
Handled Exceptions: Example

```
CREATE PROCEDURE add_department(  
    p_name VARCHAR2, p_mgr NUMBER, p_loc NUMBER) IS  
BEGIN  
    INSERT INTO DEPARTMENTS (department_id,  
        department_name, manager_id, location_id)  
    VALUES (DEPARTMENTS_SEQ.NEXTVAL, p_name, p_mgr, p_loc);  
    DBMS_OUTPUT.PUT_LINE('Added Dept: ' || p_name);  
EXCEPTION  
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE('Err: adding dept: ' || p_name);  
END;
```

```
CREATE PROCEDURE create_departments IS  
BEGIN  
    add_department('Media', 100, 1800);  
    add_department('Editing', 99, 1800);  
    add_department('Advertising', 101, 1800);  
END;
```

```
Added Dept: Media  
Err: adding dept: Editing  
Added Dept: Advertising
```


Exceptions Not Handled



Exceptions Not Handled: Example

```
SET SERVEROUTPUT ON
CREATE PROCEDURE add_department_noex(
    p_name VARCHAR2, p_mgr NUMBER, p_loc NUMBER) IS
BEGIN
    INSERT INTO DEPARTMENTS (department_id,
        department_name, manager_id, location_id)
    VALUES (DEPARTMENTS_SEQ.NEXTVAL, p_name, p_mgr, p_loc);
    DBMS_OUTPUT.PUT_LINE('Added Dept: ' || p_name);
END;
```

```
CREATE PROCEDURE create_departments_noex IS
BEGIN
    add_department_noex('Media', 100, 1800);
    add_department_noex('Editing', 99, 1800);
    add_department_noex('Advertising', 101, 1800);
END;
```

De fout wordt niet opgevangen in called program noch in calling program
→ rollback van alle DML-statements

Handled Exceptions: Example

```
SET SERVEROUTPUT ON
CREATE PROCEDURE add_department_noex(
    p_name VARCHAR2, p_mgr NUMBER, p_loc NUMBER) IS
BEGIN
    INSERT INTO DEPARTMENTS (department_id,
        department_name, manager_id, location_id)
    VALUES (DEPARTMENTS_SEQ.NEXTVAL, p_name, p_mgr, p_loc);
    DBMS_OUTPUT.PUT_LINE('Added Dept: ' || p_name);
END;
```

```
CREATE OR REPLACE PROCEDURE create_departments_noex
IS
BEGIN
    add_department_noex('Media', 100, 1800);
    add_department_noex('Editing', 99, 1800);
    add_department_noex('Advertising', 101, 1800);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('err: adding dept');
END;
```

```
SQL> exec create_departments_noex
Added Dept: Media
err: adding dept
PL/SQL procedure successfully completed.
```

De fout wordt niet opgevangen in called program
maar wel in calling program
→ GEEN rollback

Omgaan met exceptions

Een fout die optreedt tijdens de uitvoering van je code = exception
Het BEGIN-blok wordt onmiddellijk verlaten

- als exception niet wordt opgevangen in called en calling program:
 - rollback called program
 - rollback calling program
- als exception niet wordt opgevangen in called program en wel in calling program:
 - Uitgevoerde DML's called program + uitgevoerde DML's calling program (voordat called program met fout werd aangeroepen) → blijven uitgevoerd → GEEN rollback
- als exception wel wordt opgevangen in called program:
 - GEEN rollback in called program
 - verdere uitvoering code calling program

Exception handlers kan men gebruiken om bij het optreden van fouten:

- eigen foutmeldingen af te drukken
- een reeks alternatieve commando's te laten uitvoeren

Trapping Exceptions

Syntax:

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]
```

Bij functions, procedures en triggers kan na het BEGIN-block een EXCEPTION-blok gedefinieerd worden

Guidelines for Trapping Exceptions

- The `EXCEPTION` keyword starts the exception-handling section.
- Several exception handlers are allowed.
- Only one handler is processed before leaving the block.
- `WHEN OTHERS` is the last clause.

Exception Types

- Predefined Oracle server
 - Non-predefined Oracle server
- } Implicitly raised
- User-defined
- Explicitly raised

Predefined Oracle exceptions: voorbeeld

```
CREATE OR REPLACE FUNCTION optellen  
(p_getal1 NUMBER, p_getal2 NUMBER)  
RETURN NUMBER  
AS  
    v_resultaat NUMBER(2);  
BEGIN  
    v_resultaat := p_getal1 + p_getal2;  
    RETURN v_resultaat;  
END;  
/
```



Als het resultaat groter is dan 99
→ ORA-06502 numeric or value error

Predefined Oracle exceptions

Voorgedefinieerd → Oracle heeft er al een naam aan toegekend

Een lijst van voorgedefinieerde exceptions vind je in:

\$ORACLE_HOME\rdbms\admin\stdspec.sql

Concreet:

C:\oraclexe\app\oracle\product\11.2.0\server\rdbms\admin\stdspec.sql

Vb	NO_DATA_FOUND	ORA-01403
	TOO_MANY_ROWS	ORA-01422
	ZERO_DIVIDE	ORA-01476
	VALUE_ERROR	ORA-06502
	DUP_VAL_ON_INDEX	ORA-00001

Predefined Oracle exceptions: voorbeeld


```
CREATE OR REPLACE FUNCTION optellen
(p_getal1 NUMBER, p_getal2 NUMBER)
RETURN NUMBER
AS
    v_resultaat NUMBER(2);
BEGIN
    v_resultaat := p_getal1 + p_getal2;
    RETURN v_resultaat;
EXCEPTION
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('het eindresultaat is te groot');
        RETURN NULL;
END;
```

/

Predefined Oracle exceptions: voorbeeld

OPGELET: runtime-fouten bij de declaratie worden NIET opgevangen!

```
CREATE OR REPLACE FUNCTION optellen  
(p_getal1 NUMBER, p_getal2 NUMBER)  
RETURN NUMBER  
AS  
    v_resultaat NUMBER(2) := p_getal1 + p_getal2;  
BEGIN  
    RETURN v_resultaat;  
EXCEPTION  
    WHEN VALUE_ERROR THEN  
        DBMS_OUTPUT.PUT_LINE('het eindresultaat is te groot');  
        RETURN NULL;  
END;  
/
```



Non-Predefined Oracle exceptions: voorbeeld

```
CREATE OR REPLACE PROCEDURE add_country  
(p_country_id IN countries.country_id%TYPE,  
 p_country_name IN countries.country_name%TYPE,  
 p_region_id IN countries.region_id%TYPE)  
AS  
BEGIN  
  INSERT INTO countries VALUES (p_country_id, p_country_name, p_region_id);  
END;  
/
```



Bij een onbestaand regio_id
→ ORA-02291 integrity constraint violated – parent key not found

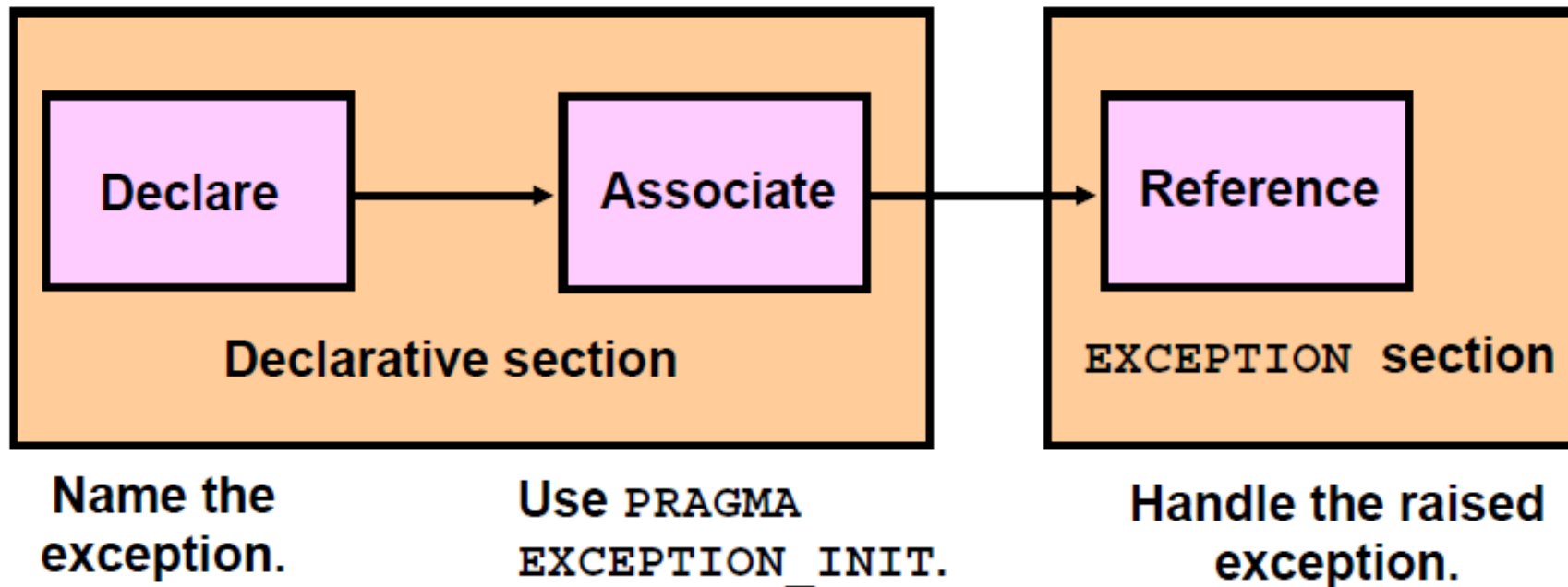
Non-Predefined Oracle exceptions

Niet-voorgedefinieerd

- ORA-02291: niet in lijst voorgedefinieerde exceptions (stdspec.sql)
- Zelf de exception declareren

1. De exception declareren en aldus een naam geven
2. De exception koppelen aan de ORA-foutcode
3. De exception afhandelen in de exception-handler

Trapping Non-Predefined Oracle Server Errors



Non-Predefined Oracle exceptions: voorbeeld

```
CREATE OR REPLACE PROCEDURE add_country
(p_country_id  IN countries.country_id%TYPE,
 p_country_name IN countries.country_name%TYPE,
 p_region_id   IN countries.region_id%TYPE)
AS
    e_foute_regio      EXCEPTION;                                = stap 1
    PRAGMA EXCEPTION_INIT(e_foute_regio, -2291);                 = stap 2
BEGIN
    INSERT INTO countries VALUES (p_country_id, p_country_name, p_region_id);
EXCEPTION                                                         = stap3
    WHEN e_foute_regio THEN
        DBMS_OUTPUT.PUT_LINE('Deze regio bestaat niet');
END;
```

Non-Predefined Error

To trap Oracle server error number -01400
("cannot insert NULL"):

```
DECLARE
  e_insert_excep EXCEPTION; ← ①
  PRAGMA EXCEPTION_INIT(e_insert_excep, -01400); ← ②
BEGIN
  INSERT INTO departments
    (department_id, department_name) VALUES (280, NULL);
EXCEPTION
  WHEN e_insert_excep THEN ← ③
    DBMS_OUTPUT.PUT_LINE('INSERT OPERATION FAILED');
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

[Geen titel]

```
anonymous block completed
INSERT OPERATION FAILED
ORA-01400: cannot insert NULL into ("ORA41"."DEPARTMENTS"."DEPARTMENT_NAME")
```

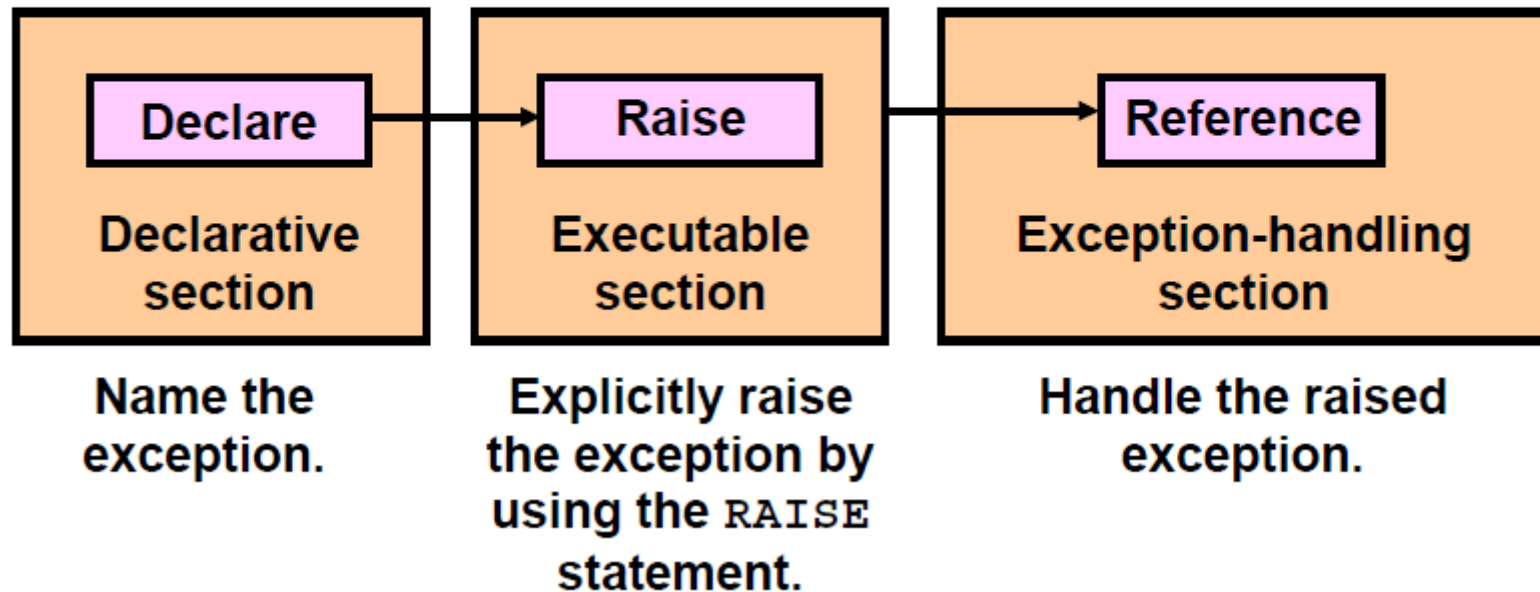

Functions for Trapping Exceptions

- **SQLCODE:** Returns the numeric value for the error code
- **SQLERRM:** Returns the message associated with the error number

SQLCODE Values: Examples

SQLCODE Value	Description
0	No exception encountered
1	User-defined exception
+100	NO_DATA_FOUND exception
<i>negative number</i>	Another Oracle server error number

Trapping User-Defined Exceptions



Het gaat hier om fouten die Oracle niet als fout ziet, maar wij als gebruiker wel.
Deze worden dus niet automatisch geraised!

User-defined exceptions: voorbeeld

```
CREATE OR REPLACE PROCEDURE update_depname  
(p_department_id IN departments.department_id%TYPE,  
 p_department_name IN departments.department_name%TYPE)  
AS  
BEGIN  
    UPDATE departments  
    SET department_name = p_department_name  
    WHERE department_id = p_department_id;  
END;  
/
```

Bij een onbestaand department_id zal er geen fout optreden!
Er zal simpelweg geen wijziging gebeuren.

User-defined exceptions: voorbeeld

```
CREATE OR REPLACE PROCEDURE update_depname  
(p_department_id IN departments.department_id%TYPE,  
 p_department_name IN departments.department_name%TYPE)  
AS
```

```
  e_onbestaand_departement EXCEPTION;
```

= stap 1

```
BEGIN
```

```
  UPDATE departments
```

```
    SET department_name = p_department_name
```

```
    WHERE department_id = p_department_id;
```

```
  IF SQL%NOTFOUND THEN
```

```
    RAISE e_onbestaand_departement;
```

= stap 2

```
  END IF;
```

```
EXCEPTION
```

= stap 3

```
  WHEN e_onbestaand_departement THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Dit department_id bestaat niet');
```

```
END;
```

RAISE_APPLICATION_ERROR Procedure

```
DECLARE
    e_name EXCEPTION;
BEGIN
    ...
    DELETE FROM employees
    WHERE last_name = 'Higgins';
    IF SQL%NOTFOUND THEN RAISE e_name;
    END IF;
EXCEPTION
    WHEN e_name THEN
        RAISE_APPLICATION_ERROR (-20999, 'This is not a valid
last name');    ...
END;
/
```

RAISE_APPLICATION_ERROR Procedure

Syntax:

```
raise_application_error (error_number,  
                        message[, {TRUE | FALSE}]);
```

- You can use this procedure to issue user-defined error messages from stored subprograms.
- You can report errors to your application and avoid returning unhandled exceptions.

<i>error_number</i>	Is a user-specified number for the exception between -20,000 and -20,999
<i>message</i>	Is the user-specified message for the exception; is a character string up to 2,048 bytes long
TRUE FALSE	Is an optional Boolean parameter (If TRUE, the error is placed on the stack of previous errors. If FALSE, which is the default, the error replaces all previous errors.)

Foutmelding: RAISE_APPLICATION_ERROR of DBMS_OUTPUT.PUT_LINE ?

- DBMS_OUTPUT.PUT_LINE
 - = melding in SQL*Plus !!
 - via andere omgeving PL/SQL-blok starten
→ geen melding
 - verschijnt alleen als SERVEROUTPUT is ingeschakeld
 - programma wordt niet afgebroken, maar eindigt correct
- RAISE_APPLICATION_ERROR
 - geeft foutmelding
ook zichtbaar als PL/SQL-prog vanuit andere applicatie w opgestart
 - niet afhankelijk van SERVEROUTPUT
 - zorgt dat programma direct wordt afgebroken
 - zorgt voor rollback van wijzigingen die het prog heeft doorgevoerd

RAISE_APPLICATION_ERROR Procedure

- Used in two different places:
 - Executable section
 - Exception section
- Returns error conditions to the user in a manner consistent with other Oracle server errors

Soms wil je een fout opvangen en enkele acties uitvoeren, maar je wil alsnog eindigen in een foutsituatie om dit door te geven aan de achterliggende applicatie

→ in de exception-handler gebruik je RAISE_APPLICATION_ERROR

RAISE_APPLICATION_ERROR Procedure

Executable section:

```
BEGIN
...
DELETE FROM employees
WHERE manager_id = v_mgr;
IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20202,
        'This is not a valid manager');
END IF;
...
```

Exception section:

```
...
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR (-20201,
            'Manager is not a valid employee.');
```

END;
/

WHEN OTHERS

- correspondeert niet met een specifieke fout
- OTHERS is steeds de laatste handle
- GEVOLG:
 - alle fouten worden opgevangen + prog eventueel succesvol afgesloten eventueel aan het einde zelf een foutsituatie creëren, zodat prog wordt afgebroken en wijzigingen worden teruggedraaid.
Hoe?
`RAISE exception_naam;`
`RAISE;` -- re-raise de huidige exception

WHEN OTHERS: voorbeeld

```
CREATE OR REPLACE PROCEDURE update_depname
(p_department_id IN departments.department_id%TYPE,
 p_department_name IN departments.department_name%TYPE)
AS
    e_onbestaand_departement          EXCEPTION;
    v_melding                         VARCHAR2(60);
BEGIN
    UPDATE departments
    SET department_name = p_department_name
    WHERE department_id = p_department_id;
    IF SQL%NOTFOUND THEN
        RAISE e_onbestaand_departement;
    END IF;
EXCEPTION
    WHEN e_onbestaand_departement THEN
        DBMS_OUTPUT.PUT_LINE('Dit department_id bestaat niet');
    WHEN OTHERS THEN
        v_melding := 'Fout '||SQLCODE||' is opgetreden, met melding '||SQLERRM;
        INSERT INTO log_table
        VALUES (0, v_melding, sysdate);
        COMMIT;  --niet altijd nodig; hangt af van calling program
        RAISE;
END;
```

Vb netwerkfout

WHEN OTHERS: voorbeeld

```
CREATE OR REPLACE PROCEDURE testproc
AS
    v_last employees.last_name%TYPE;
    v_mess varchar2(100);
BEGIN
    SELECT last_name
    INTO v_last
    FROM employees
    WHERE first_name = 'John';
EXCEPTION
    WHEN OTHERS THEN
        v_mess := sqlerrm;
        DBMS_OUTPUT.PUT_LINE('FOUT!!!' || sqlcode || ' - ' || sqlerrm);
        INSERT INTO log_tabel VALUES(sysdate, null, v_mess);
        RAISE;
END;
```

```
BEGIN
    testproc;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('er zijn
        meerdere mensen met de
        voornaam John');
END;
/
```

Samenvatting

Soort exception	Opgeworpen door	De naam wordt gegeven door	Heeft altijd een errorcode
Predefined Oracle-exception	Oracle	Oracle	Ja
Non-predefined Oracle exception	Oracle	Gebruiker	Ja
User-defined exception	Gebruiker	Gebruiker	Nee