Kivételkezelés

Kivételkezelés – 1

- Futási időben bekövetkező hibák
  - beépített (futtató rendszer váltja ki):
    - előre definiált
    - nem előre definiált
  - felhasználói
- Az előre definiált és a felhasználói kivételeknek van nevük
- Minden kivételnek van kódja és szövege
- Két beépített függvény:
  - `SQLCODE`: a legutoljára bekövetkezett kivétel kódja
  - `SQLERRM[(`*k*`)]`: a legutoljára bekövetkezett (ill. a *k* kódú) kivételhez tartozó szöveg (hibaüzenet)

Kivételkezelés – 2

- `SQLCODE` lehetséges értékei:

  0, ha nem történt kivétel,

  1, ha felhasználói kivétel történt,

  +100, ha a NO_DATA_FOUND beépített kivétel következett be,

  negatív szám, bármely más beépített kivétel esetén.

Kivételnév	Hibakód	Leírás
ACCESS_INTO_NULL	ORA-06530	Attempted to assign values to the attributes of an uninitialized object
CASE_NOT_FOUND	ORA-06592	None of the choices in the WHEN clauses of a CASE statement are selected, and there is no ELSE clause.
COLLECTION_IS_NULL	ORA-06531	Attempted to apply collection methods other than EXISTS to an uninitialized nested table or VARRAY
CURSOR_ALREADY_OPEN	ORA-06511	Attempted to open an already-open cursor
DUP_VAL_ON_INDEX	ORA-00001	Attempted to insert a duplicate value
INVALID_CURSOR	ORA-01001	Illegal cursor operation occurred.
INVALID_NUMBER	ORA-01722	Conversion of character string to number fails.
LOGIN_DENIED	ORA-01017	Logging on to the Oracle server with an invalid username or password
NO_DATA_FOUND	ORA-01403	Single row SELECT returned no data.
NOT_LOGGED_ON	ORA-01012	PL/SQL program issues a database call without being connected to the Oracle server.
PROGRAM_ERROR	ORA-06501	PL/SQL has an internal problem.
ROWTYPE_MISMATCH	ORA-06504	Host cursor variable and PL/SQL cursor variable involved in an assignment have incompatible return types.
STORAGE_ERROR	ORA-06500	PL/SQL ran out of memory, or memory is corrupted.
SUBSCRIPT_BEYOND_COUNT	ORA-06533	Referenced a nested table or VARRAY element by using an index number larger than the number of elements in the collection
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	Referenced a nested table or VARRAY element by using an index number that is outside the legal range (for example, –1)
SYS_INVALID_ROWID	ORA-01410	The conversion of a character string into a universal ROWID fails because the character string does not represent a valid ROWID.
TIMEOUT_ON_RESOURCE	ORA-00051	Time-out occurred while the Oracle server was waiting for a resource.
TOO_MANY_ROWS	ORA-01422	Single-row SELECT returned more than one row.
VALUE_ERROR	ORA-06502	Arithmetic, conversion, truncation, or size-constraint error occurred.
ZERO_DIVIDE	ORA-01476	Attempted to divide by zero

PL/SQL blokk

```
[címke]
[DECLARE
   deklarációs utasítás(ok)]
BEGIN
   végrehajtható utasítás(ok)
   [EXCEPTION
      kivételkezelő]
END  [név];
```

Kivételkezelő

- Programegység végén:

```
EXCEPTION
  WHEN kivételnév [OR kivételnév]…
     THEN utasítás [utasítás]…
  [WHEN kivételnév [OR kivételnév]…
     THEN utasítás [utasítás]…]
  [WHEN OTHERS
     THEN utasítás [utasítás]…]
```

Kivételkezelés lépései

Ha bekövetkezik egy kivétel…

- …végrehajtható részben
  - a futás félbeszakad
  - a futtató rendszer megnézi, hogy van-e a programegység végén kivételkezelő, és annak valamely `WHEN` ága nevesíti-e a bekövetkezett kivételt
    - ha igen, lefutnak a `THEN` utáni utasítások
    - különben, ha van kivételkezelő, és abban `WHEN OTHERS` ág, akkor az ottani utasítások futnak le
    - különben a kivétel továbbadódik az aktiváló környezetnek
      - ha ilyen nincs, akkor `UNHANDLED_EXCEPTION` váltódik ki
- …deklarációs részben vagy kivételkezelőben
  - azonnal továbbadódik

```
CREATE OR REPLACE PROCEDURE select_item
   ( t_column VARCHAR2, t_name VARCHAR2 ) AUTHID DEFINER IS
  temp VARCHAR2(30);
 BEGIN
  temp := t_column;
  SELECT COLUMN_NAME INTO temp
   FROM USER_TAB_COLS
   WHERE TABLE_NAME=UPPER(t_name) AND
   COLUMN_NAME=UPPER(t_column);
  temp := t_name;
  SELECT OBJECT_NAME INTO temp
   FROM USER_OBJECTS
   WHERE OBJECT_NAME = UPPER(t_name) AND OBJECT_TYPE = 'TABLE';
 EXCEPTION
   WHEN NO_DATA_FOUND
    THEN DBMS_OUTPUT.PUT_LINE ('No Data found for SELECT on ' 
         || temp);
   WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE ('Unexpected error');
END;
/
```

Előre definiált kivételek kezelése

- A kivételkezelőben hivatkozunk a nevére

```
DECLARE
   v_id hr.employees.employee_id%TYPE;
BEGIN
   SELECT employee_id
     INTO v_id
     FROM hr.employees
    WHERE last_name='John';

EXCEPTION
   WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Nincs egyetlen John sem.');
   WHEN TOO_MANY_ROWS THEN
       DBMS_OUTPUT.PUT_LINE('Több John is van.');
END;
/
```

Nem előre definiált kivételek kezelése

- Deklarációs részben
  - deklarálunk egy kivételnevet
  - összerendeljük egy kóddal
- A kivételkezelőben hivatkozunk a nevére

(Megjegyzés: `ORA-01400: cannot insert NULL`)

```
DECLARE
   beszur_kivetel EXCEPTION;
   PRAGMA EXCEPTION_INIT(beszur_kivetel, -1400);
BEGIN
   INSERT INTO departments(department_id, department_name)
       VALUES (999, NULL);
EXCEPTION
   WHEN beszur_kivetel THEN
       DBMS_OUTPUT.PUT_LINE('Sikertelen beszúrás!');
       DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

Felhasználói kivételek kezelése

- Deklarációs részben deklarálunk egy kivételnevet
- Végrehajtható részben kiváltjuk a kivételt (`RAISE`)
- A kivételkezelőben hivatkozunk a nevére

```
VARIABLE elvart NUMBER
EXECUTE :elvart := 130

DECLARE
   keves_dolgozo EXCEPTION;
   v_db  PLS_INTEGER;
BEGIN
   SELECT COUNT(*) INTO v_db FROM hr.employees;
   IF v_db < :elvart THEN RAISE keves_dolgozo; END IF;
EXCEPTION
   WHEN keves_dolgozo THEN
        DBMS_OUTPUT.PUT_LINE('Több dolgozóra lenne szükség!');
END;
/
```

A `RAISE_APPLICATION_ERROR` eljárás

- Három paramétere van:
  - egy -20000 és -20999 közé eső szám (a kivétel kódja),
  - egy maximum 2048 bájt hosszúságú sztring (a kivétel szövege),
  - egy opcionális logikai érték
    - `TRUE`: a kivétel az eddigi hibák vermének a tetejére kerül,
    - `FALSE`: kiürül a verem, és csak ez a kód kerül bele (alapértelmezés).
- Az eljárás meghívásakor kiváltódik a megadott kódú felhasználói kivétel a megadott üzenettel.
- Előnyök:
  - tárolt alprogramok esetén
  - alkalmazásunk számára tudunk információt biztosítani a hibáról

```
CREATE FUNCTION get_fizetes
  (p_id HR.EMPLOYEES.EMPLOYEE_ID%TYPE)
RETURN NUMBER IS
  v HR.EMPLOYEES.SALARY%TYPE;
  atlag v%TYPE;
  tul_alacsony_fizu EXCEPTION;
BEGIN
  SELECT SALARY INTO v FROM
  HR.EMPLOYEES WHERE employee_id=p_id;
  SELECT AVG(SALARY) INTO atlag
  FROM HR.EMPLOYEES;
  IF v/atlag < 0.5 THEN
    RAISE tul_alacsony_fizu;
  END IF;
  RETURN v;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20100,
      'Nincs ' || p_id ||
      ' azonosítójú alkalmazott!');
 WHEN tul_alacsony_fizu THEN
    RAISE_APPLICATION_ERROR(-20101,
      'A ' || p_id ||
      ' azonosítójú alkalmazott' ||
   ' fizetése túlságosan alacsony!');
END get_fizetes;
/
```

```
DECLARE
  hibas_alk_azon EXCEPTION;
  kis_penz EXCEPTION;
  PRAGMA EXCEPTION_INIT
    (hibas_alk_azon, -20100);
  PRAGMA EXCEPTION_INIT
    (kis_penz, -20101);
BEGIN
  FOR i IN (110, 210) LOOP
    BEGIN
      DBMS_OUTPUT.PUT_LINE
        (i || ' fizetése: '
           || get_fizetes(i));
    EXCEPTION
      WHEN hibas_alk_azon THEN
        DBMS_OUTPUT.PUT_LINE
          (SQLERRM);
    END;
  END LOOP;

EXCEPTION
  WHEN kis_penz THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

```
DROP TABLE employees_temp;
CREATE TABLE employees_temp AS
SELECT employee_id, salary, commission_pct FROM employees;

DECLARE
  sal_calc NUMBER(8,2);
BEGIN
 INSERT INTO employees_temp (employee_id, salary,
  commission_pct)
 VALUES (301, 2500, 0);
 SELECT (salary / commission_pct) INTO sal_calc
  FROM employees_temp
  WHERE employee_id = 301;
 INSERT INTO employees_temp
  VALUES (302, sal_calc/100, .1);
 DBMS_OUTPUT.PUT_LINE('Row inserted.');
EXCEPTION
  WHEN ZERO_DIVIDE
   THEN DBMS_OUTPUT.PUT_LINE('Division by zero.');
END;
/
```

```
DECLARE
 sal_calc NUMBER(8,2);
BEGIN
 INSERT INTO employees_temp (employee_id, salary, commission_pct)
 VALUES (301, 2500, 0);

 BEGIN
   SELECT (salary / commission_pct) INTO sal_calc
   FROM employees_temp WHERE employee_id = 301;
 EXCEPTION WHEN ZERO_DIVIDE
  THEN DBMS_OUTPUT.PUT_LINE
       ('Substituting 2500 for undefined number.');
   sal_calc := 2500;
 END;

 INSERT INTO employees_temp VALUES (302, sal_calc/100, .1);
 DBMS_OUTPUT.PUT_LINE('Enclosing block: Row inserted.');
EXCEPTION WHEN ZERO_DIVIDE
 THEN DBMS_OUTPUT.PUT_LINE('Enclosing block: Division by zero.');
END;
/
```

```
DROP TABLE results;
CREATE TABLE results
 ( res_name VARCHAR(20),
   res_answer VARCHAR2(3) );
CREATE UNIQUE INDEX res_name_ix ON results (res_name);
INSERT INTO results (res_name, res_answer)
  VALUES ('SMYTHE', 'YES');
INSERT INTO results (res_name, res_answer)
  VALUES ('JONES', 'NO');
```

```
DECLARE
 name VARCHAR2(20) := 'SMYTHE'; answer VARCHAR2(3) := 'NO';
 suffix NUMBER := 1;
BEGIN
 FOR i IN 1..5 LOOP -- Try transaction at most 5 times.
 DBMS_OUTPUT.PUT('Try #' || i);
 BEGIN -- sub-block begins
  SAVEPOINT start_transaction; -- transaction begins
  DELETE FROM results WHERE res_answer = 'NO';
  INSERT INTO results (res_name,res_answer) VALUES (name,answer);
  COMMIT; DBMS_OUTPUT.PUT_LINE(' succeeded.');
  EXIT;
 EXCEPTION
  WHEN DUP_VAL_ON_INDEX
    THEN DBMS_OUTPUT.PUT_LINE(' failed; trying again.');
         ROLLBACK TO start_transaction; -- Undo changes.
         suffix := suffix + 1; -- Try to fix problem.
         name := name || TO_CHAR(suffix);
 END; -- sub-block ends
 END LOOP;
END;
```