**Obsługa błędów**

Procedury składowane umożliwiają również obsługę błędów.

**Składnia DECLARE HANDLER**

|  |  |
| --- | --- |
|  | DECLARE { EXIT | CONTINUE }  HANDLER FOR  { kod\_błędu | { SQLSTATE znak\_błędu } | warunek }  wyrażenie SQL |

Obsługa błędu polega na napisaniu kodu procedury, który uruchomi się w odpowiednim momencie. Wyróżniamy dwa przypadki:

* EXIT – który powoduje opuszczenie procedury tuż po próbie wykonania polecenia, która właśnie zwróci nam odpowiedni błąd,
* CONTINUE – który nie spowoduje przerwania działania procedury, pozostałe instrukcje procedury wykonają się w całości

**Przykład DECLARE EXIT HANDLER**

CREATE PROCEDURE proc01 (parametr INT)  
 BEGIN  
    DECLARE EXIT HANDLER FOR 1216  
    INSERT INTO tabela1 VALUES  
     (CONCAT('Data: ',now(),  
     '. Foreign Key Reference Failure For  
     Value = ',parametr));  
    INSERT INTO tabela2 VALUES (parametr);  
  END;//

**Przykład DECLARE CONTINUE HANDLER**

|  |  |
| --- | --- |
|  | CREATE PROCEDURE proc02()   BEGIN     DECLARE CONTINUE HANDLER     FOR SQLSTATE '23000' SET @x = 1;     SET @y = 1;     INSERT INTO tabela(x) VALUES (1);     SET @y = 2;  INSERT INTO tabela(x) VALUES (1);     SET @x = 3;   END;// |
|  |  |

**Przykład uwzględniający kodowanie znaków**

CREATE TABLE tabela (x smallint unsigned not null primary key,y int unsigned);

CREATE TABLE bledy(error varchar(100)) CHARSET cp1250;

CREATE PROCEDURE nazwa\_procedury(IN k INT, IN w INT)

BEGIN

/\* Pierwszy handler \*/

DECLARE EXIT HANDLER FOR 1062

BEGIN

SET CHARSET cp1250;

INSERT INTO bledy VALUES (CONCAT(’Data: ’,now(),’ powtórzony klucz ’,k));

END;

/\* Drugi handler \*/

DECLARE EXIT HANDLER FOR 1264

BEGIN

SET CHARSET cp1250;

INSERT INTO bledy VALUES (CONCAT(’Data: ’,now(),’ ujemna wartość ’,w));

END;

/\* Ciało procedury - wstawianie \*/

INSERT INTO tabela2 VALUES (k, w);

END; //