Error Handling and Exceptions in MySQL

- Understanding handlers, types, and examples

Instructor : Sriya Ivaturi

Introduction to Error Handling

- - Error handling lets you manage runtime issues
- - MySQL uses HANDLERs inside stored programs

Why Handle Errors?

- - Prevents unexpected termination
- - Helps in debugging
- - Returns meaningful messages
- - Maintains application stability

Syntax of DECLARE HANDLER

- DECLARE handler_action HANDLER
- FOR condition_value
- BEGIN
- -- handler logic
- END;

Types of HANDLER Actions

- CONTINUE – skip the error and continue
- EXIT – exit the block/procedure
- UNDO – rollback changes (Not supported in MySQL)

Types of Condition Values

- SQLSTATE value: '23000' → Integrity error
- MySQL error #: 1062 → Duplicate entry
- Predefined: NOT FOUND, SQLEXCEPTION, SQLWARNING

CONTINUE Handler Example

- DECLARE CONTINUE HANDLER FOR NOT FOUND
- SET done = TRUE;

EXIT Handler Example

- DECLARE EXIT HANDLER FOR SQLEXCEPTION
- BEGIN
- ROLLBACK;
- SELECT 'Error occurred. Transaction cancelled.';
- END;

Custom Condition & HANDLER

- DECLARE insufficient_balance CONDITION FOR SQLSTATE '45000';
- DECLARE EXIT HANDLER FOR insufficient_balance
- BEGIN
- SELECT 'Balance too low!';
- END;

SIGNAL for Custom Errors

- IF NEW.amount < 0 THEN
- SIGNAL SQLSTATE '45000'
- SET MESSAGE_TEXT = 'Amount cannot be negative';
- END IF;

Error Handling in Functions

- CREATE FUNCTION safe_divide(a INT, b INT)
- RETURNS DECIMAL(10,2)
- BEGIN
- DECLARE EXIT HANDLER FOR SQLEXCEPTION RETURN 0;
- RETURN a / b;
- END;

Tricky Questions

- Q1: Can we declare multiple handlers?
- Q2: Difference between CONTINUE and EXIT?
- Q3: Can a handler contain SIGNAL?

Tricky Answers

- A1: Yes, for different errors
- A2: CONTINUE → resume
- EXIT → leave block
- A3: Yes, SIGNAL can raise custom errors

Summary

- - Use DECLARE HANDLER to catch errors
- - Types: CONTINUE, EXIT
- - Combine with SIGNAL for custom errors

Ex: Safe Division in Function

```
CREATE FUNCTION safe_divide(a INT, b INT)
RETURNS DECIMAL(10,2)
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION RETURN 0;
  RETURN a / b;
END;
```

Ex: NOT FOUND in Cursors

- DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
- OPEN cur;
- LOOP FETCH cur INTO var1;
- IF done THEN LEAVE;
- END LOOP;
- CLOSE cur;

Ex: Rollback on Error

- DECLARE EXIT HANDLER FOR SQLEXCEPTION
- BEGIN ROLLBACK;
- SELECT 'Error occurred';
- END;
- START TRANSACTION;
- UPDATE accounts ...;
- COMMIT;

Ex: SIGNAL in Triggers

- IF NEW.salary < 0 THEN
- SIGNAL SQLSTATE '45000'
- SET MESSAGE_TEXT = 'Negative salary not allowed';
- END IF;

Ex: Duplicate Entry Handler

- DECLARE EXIT HANDLER FOR 1062
- BEGIN
- SELECT 'User exists';
- END;
- INSERT INTO users(username) VALUES ('abc');

Ex: Multiple Handlers

- DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
- DECLARE EXIT HANDLER FOR SQLEXCEPTION
- BEGIN
- ROLLBACK;
- SELECT 'Error occurred';
- END;

Ex: SQLWARNING Handler

- DECLARE CONTINUE HANDLER FOR SQLWARNING
- BEGIN
- SET @warning = 'Something might be wrong!';
- END;