DBMSL

Roll No : 31147

Assignment 5

Date of Completion : 12/9/20. Date of submission : 12/11/20.

Problem Statement :

Write a PL/SQL block of code for the following requirement

Schema :-

Borrower (Cust_id, Name, Date of payment, Name of Schema).

Fine (Cust_id, Date, Amt)

1) Accept cust_id and name of scheme from user.
2) Check the number of days (from date of payment, if days are between 15 to 30 then fine amount will be Rs 5 per day.
3) If no of days > 30 the fine will be 50 per day and for days less than 30, 5 Rs per day.
4) If payment status will change from 'I' to 'R'
5) If condition is true, then details will be stored into fine table.

Objective And Outcomes :

1) Understand the Exception handling in Procedures
2) Learn to use procedure in MySQL
3) Should be able to understand the Exceptions in PL/SQL.

Theory :

What is PL/SQL ?

PL/SQL is a combination of SQL along with procedural features of programming language. It was developed by Oracle Coporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of the three key programming languages embedded in the Oracle database, along with SQL itself & Java.

PL/SQL is defined by the keywords DECLARE, BEGIN, EXCEPTION, END.

1) Declarative statements that declare variables constants and other code elements, which can be used within that block.
2) Executable: Statements that run when block is executed.
3) Exception Handling: A specially structured section you can use to catch any exception that are raised when the executables run into error.

Page No.
Date

Exception Handling:

When an error occurs inside a stored procedure, it is important to handle it appropriately such as continuing or exiting the current code blocks execution and issuing a meaningfull error message.

MySQL provides an easy way to define handlers that handle from general conditions such as warnings or exceptions to specific conditions.

Declare a handler:

To declare a handler you use the DECLARE HANDLE statement as follows:

DECLARE action HANDLER FOR condition_value statement

If condition whose value matches the condition value MySQL will execute the statement and continue or exit the current code block based on the action.

The action accepts one of the following value

i) CONTINUE.

ii) EXIT

Condition_value specifies a particular condition or a class of condition that activate the handler. The condition_value accept one of the following values:

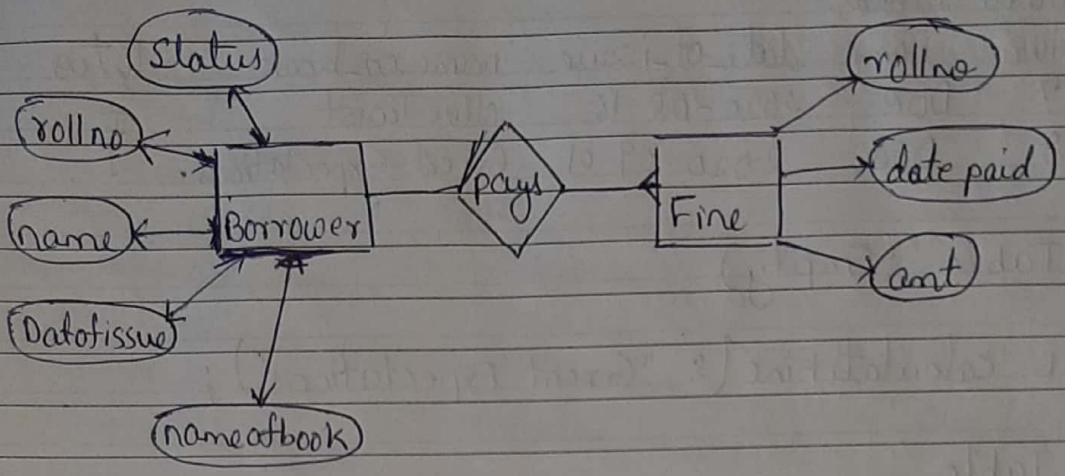
SQL error code

A standard SQL state value. Or it can be an SQL warning, NOT FOUND or SQLEXCEPTION condition, which is shorthand for the class of SQLSTATE values.

Examples for MySQL error handling

i) DECLARE CONTIVE HANDLER FOR SQLEXCEPTION SELECT "Error";

ER/DIAGRAM:



Program Listing:

```
DELIMITER //

CREATE PROCEDURE calculateFine(IN roll_no INT, IN title VARCHAR(100))
BEGIN
DECLARE no_of_days INT;
DECLARE date_issued DATE;
DECLARE fineAmount INT;
DECLARE EXIT HANDLER FOR SQLEXCEPTION SELECT "Error has occurred. Exception occured";
SELECT date_of_issue INTO date_issued
FROM borrower
WHERE rollno=roll_no AND name_of_book=title;
SELECT DATEDIFF(CURDATE(),date_issued) INTO no_of_days;
IF no_of_days BETWEEN 15 AND 30 THEN
SET fineAmount = 5*no_of_days;
INSERT INTO fine VALUES(roll_no, CURDATE(), fineamount);
ELSEIF no_of_days > 30 THEN
SET fineAmount = 50* no_of_days;
INSERT INTO fine VALUES(roll_no, CURDATE(), fineAmount);
END IF;
UPDATE borrower
SET Status = "R"
WHERE rollno = roll_no AND name_of_book = title;
END;
//
```

Page No.
Date

Test Cases:

Borrower Table.

rollno	name	date_of_issue	name_of_book	status
7	Doe	2020-08-16	Olive Twist	I
8	Doe	2020-09-01	Great Expectations.	I.

Fine Table (Empty)

1) CALL calculateFine(8, "Great Expectations");

Fine Table

rollno	date_paid	amt.
8	2020-09-22	105.

2) CALL calculateFine(7, "Olive Twist");

Fine Table

rollno	date_paid	amt.
8	2020-09-22	105
7	2020-09-22	1850.

BorrowerTable.

rollno	name	date-of-issue	name_of_book	status
7	Doe	2020-08-16	Oliver Twist	R
8	Doe	2020-09-01	Great Expectation	R.

Conclusion:

Successfully understood and implemented stored procedures and Exception handling in the assignment to calculate Fine for a book borrower.

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM borrower;
+--------+-----------+---------------+--------------------+--------+
| rollno | name      | date_of_issue | name_of_book       | status |
+--------+-----------+---------------+--------------------+--------+
|      1 | Pradyumna | 2020-09-12    | Harry Potter 1     | I      |
|      2 | Shubham   | 2020-09-13    | Harry Potter 2     | I      |
|      3 | Pranav    | 2020-09-14    | Harry Potter 3     | I      |
|      4 | John      | 2020-08-29    | Harry Potter 1     | I      |
|      5 | Will      | 2020-07-29    | Harry Potter 2     | I      |
|      6 | Mickey    | 2020-08-15    | Harry Potter 3     | I      |
|      7 | Doe       | 2020-08-16    | Oliver Twist       | I      |
|      8 | Doe       | 2020-09-01    | Great Expectations | I      |
+--------+-----------+---------------+--------------------+--------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM fine;
Empty set (0.00 sec)

mysql> CALL calculateFine(1,"Harry Potter 1");
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM borrower;
+--------+-----------+---------------+--------------------+--------+
| rollno | name      | date_of_issue | name_of_book       | status |
+--------+-----------+---------------+--------------------+--------+
|      1 | Pradyumna | 2020-09-12    | Harry Potter 1     | R      |
|      2 | Shubham   | 2020-09-13    | Harry Potter 2     | I      |
|      3 | Pranav    | 2020-09-14    | Harry Potter 3     | I      |
|      4 | John      | 2020-08-29    | Harry Potter 1     | I      |
|      5 | Will      | 2020-07-29    | Harry Potter 2     | I      |
|      6 | Mickey    | 2020-08-15    | Harry Potter 3     | I      |
|      7 | Doe       | 2020-08-16    | Oliver Twist       | I      |
|      8 | Doe       | 2020-09-01    | Great Expectations | I      |
+--------+-----------+---------------+--------------------+--------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM fine;
+--------+------------+------+
| rollno | date_paid  | amt  |
+--------+------------+------+
|      1 | 2020-09-29 |   85 |
+--------+------------+------+
1 row in set (0.00 sec)

mysql> CALL calculateFine(1,"Harry Potter 1");
+------------------+
| Already returned |
+------------------+
| Already returned |
+--------------------+
```

MySQL 8.0 Command Line Client

```
mysql> CALL calculateFine(1,"Harry Potter 1");
+------------------+
| Already returned |
+------------------+
| Already returned |
+------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL calculateFine(2,"Harry Potter 1");
+------------------+
| RECORD NOT FOUND |
+------------------+
| RECORD NOT FOUND |
+------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL calculateFine(6,"Harry Potter 3");
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM fine;
+--------+------------+------+
| rollno | date_paid  | amt  |
+--------+------------+------+
|      1 | 2020-09-29 |   85 |
|      6 | 2020-09-29 | 2250 |
+--------+------------+------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM borrower;
+--------+-----------+---------------+--------------------+--------+
| rollno | name      | date_of_issue | name_of_book       | status |
+--------+-----------+---------------+--------------------+--------+
|      1 | Pradyumna | 2020-09-12    | Harry Potter 1     | R      |
|      2 | Shubham   | 2020-09-13    | Harry Potter 2     | I      |
|      3 | Pranav    | 2020-09-14    | Harry Potter 3     | I      |
|      4 | John      | 2020-08-29    | Harry Potter 1     | I      |
|      5 | Will      | 2020-07-29    | Harry Potter 2     | I      |
|      6 | Mickey    | 2020-08-15    | Harry Potter 3     | R      |
|      7 | Doe       | 2020-08-16    | Oliver Twist       | I      |
|      8 | Doe       | 2020-09-01    | Great Expectations | I      |
+--------+-----------+---------------+--------------------+--------+
8 rows in set (0.00 sec)

mysql>
```