Wai Yan Kaung

Prof. Gabor,

Database II

03/06/2016

## Assignment 4-3: Using Implicit Cursors

The BB\_SHOPPER table in the Brewbean's database contains a column named PROMO that specifies promotions to send to shoppers. This column needs to be cleared after the promotion has been sent. First, open the assignment04-03.txt file in a text editor (such as Notepad). Run the UPDATE and COMMIT statements at the top of this file (not the anonymous block at the end). Modify the anonymous block so that it displays the number of rows updated onscreen. Run the block.

Solution:

DECLARE

BEGIN

UPDATE bb\_shopper

SET promo = NULL

WHERE promo IS NOT NULL;

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT);

END;

Results:

4

## Assignment 4-4: Using Exception Handling

In this assignment, you test a block containing a CASE statement for errors, and then add an exception handler for a predefined exception:

1. In Notepad, open the assignment04-04.sql file. Review the block, which contains a CASE statement and no exception handlers.

2. Copy and paste the block into SQL Developer, and run the block. An error is raised because the state of NJ isn't included in the CASE statement; recall that a CASE statement must find a matching case.

3. To correct this problem, add a predefined EXCEPTION handler that addresses this error and displays "No tax" onscreen.

4. Run the block again. Your results should look like the one below. Now the error is handled in the block's EXCEPTION section.

Solution:

DECLARE

lv\_tax\_num NUMBER(2,2);

noTax EXCEPTION;

BEGIN

CASE 'NJ'

WHEN 'VA' THEN lv\_tax\_num := .04;

WHEN 'NC' THEN lv\_tax\_num := .02;

WHEN 'NY' THEN lv\_tax\_num := .06;

ELSE RAISE noTax;

END CASE;

DBMS\_OUTPUT.PUT\_LINE('tax rate = '||lv\_tax\_num);

EXCEPTION

WHEN noTax THEN

DBMS\_OUTPUT.PUT\_LINE('No tax');

END;

Results:

No tax

Statement processed.

0.01 seconds

## Assignment 4-5: Handling Predefined Exceptions

A block of code has been created to retrieve basic customer information (see the assignment04-05.sql file). The application page was modified so that an employee can enter a customer number that could cause an error. An exception handler needs to be added to the block that displays the message "Invalid shopper ID" onscreen. Use an initialized variable named lv\_shopper\_num to provide a shopper ID. Test the block with the shopper ID 99.

Solution:

DECLARE

rec\_shopper bb\_shopper%ROWTYPE;

lv\_shopper\_num bb\_shopper.idshopper%TYPE;

BEGIN

lv\_shopper\_num:= 99;

SELECT \*

INTO rec\_shopper

FROM bb\_shopper

WHERE idShopper = lv\_shopper\_num;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('invalid shopper id');

END;

Results:

invalid shopper id Statement processed.

0.01 seconds

## Assignment 4-6: Handling Exceptions with Undefined Errors

Brewbean's wants to add a check constraint on the QUANTITY column of the BB\_BASKETITEM table. If a shopper enter a quantity value greater than 20 for an item, Brewbean's wants to display the message "Check Quantity" onscreen. Using a text editor, open the assignment04-06.txt file. The first statement, ALTER TABLE, must be executed to add the check constraint. The next item is a PL/SQL block containing an INSERT action that tests this check constraint. Add code to this block to trap the check constraint violation and display the message.

Solution:

BEGIN

INSERT INTO bb\_basketitem

VALUES (88,8,10.8,21,16,2,3);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error when attempting insert:');

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

Results:

Error when attempting insert: ORA-02290: check constraint (DBEV2.BITEMS\_QTY\_CK) violated 1 row(s) inserted.

0.01 seconds

## Assignment 4-7: Handling Exceptions with User-Defined Errors

Sometimes Brewbean's customers mistakenly leave an item out of a basket that's already been checked out, so they create a new basket containing the missing items. However, they request that the baskets be combined so that they aren't charged extra shipping. An application page has been developed that enables employees to change the basket ID of items in the BB\_BASKETITEM table to another existing basket's ID to combine the baskets. A block has been constructed to support this page (see the assignment04-07.sql file). However, an exception handler needs to be added to trap the situation of an invalid basket ID being entered for the original basket. In this case, the UPDATE affects no rows but doesn't raise an Oracle error. The handler should display the message "Invalid original basket ID" onscreen. Use an initialized variable named lv\_old\_num with a value of 30 and another named lv\_new\_num with a value of 4 to provide values to the block. First, verify that no item rows with the basket ID 30 exist in the BB\_BASKETITEM table.

Solution:

DECLARE

lv\_old\_num NUMBER(3) := 30;

lv\_new\_num NUMBER(3) := 4;

invalidBasketId EXCEPTION;

BEGIN

UPDATE bb\_basketitem

SET idBasket = lv\_new\_num

WHERE idBasket = lv\_old\_num;

IF SQL%ROWCOUNT = 0 THEN

RAISE invalidBasketId;

END IF;

EXCEPTION

WHEN invalidBasketId THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid original basket ID');

END;

Reuslts:

Invalid original basket ID

0.01 seconds

## Assignment 4-8: Processing and Updating a Group of Rows

To help track employee information, a new EMPLOYEE table was added to the Brewbean's database. Review the data in this table. A PL/SQL block is needed to calculate annual raises and update employee salary amounts in the table. Create a block that addresses all the requirements in the following list. All salaries in the EMPLOYEE table are recorded as monthly amounts. Tip: Display the calculated salaries for verification before including the update action.

· Calculate 6% annual raises for all employees except the president. If a 6% raise totals more than $2,000, cap the raise at $2,000.

· Update the salary for each employee in the table.

· For each employee number, display the current annual salary, raise, and proposed new annual salary.

· Finally, following the details for each employee, show the total cost of all employees' salary increases for Brewbean's.

Solution:

DECLARE

CURSOR cursor IS

SELECT \* FROM employee

WHERE JOB != 'PRESIDENT';

annualRaise employee.sal%TYPE;

totalSalaryCost employee.sal%TYPE := 0;

record employee%ROWTYPE;

BEGIN

FOR record IN cursor LOOP

annualRaise := record.sal;

annualRaise := annualRaise \* .06;

totalSalaryCost := totalSalaryCost + annualRaise;

annualRaise := annualRaise + record.sal;

DBMS\_OUTPUT.PUT\_LINE('Employee #:' || record.empno || ' Current Salary: $' || record.sal || ' Salary Raise: $' || record.sal \* .06 || ' New Salary: $' || annualRaise);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Total of all raises combined: $' || totalSalaryCost);

END;

Results:

Employee #:7698 Current Salary: $2850 Salary Raise: $171 New Salary: $3021

Employee #:7782 Current Salary: $2450 Salary Raise: $147 New Salary: $2597

Employee #:7566 Current Salary: $2975 Salary Raise: $178.5 New Salary: $3153.5 Employee #:7654 Current Salary: $1250 Salary Raise: $75 New Salary: $1325

Employee #:7499 Current Salary: $1600 Salary Raise: $96 New Salary: $1696

Employee #:7844 Current Salary: $1500 Salary Raise: $90 New Salary: $1590

Employee #:7900 Current Salary: $950 Salary Raise: $57 New Salary: $1007

Employee #:7521 Current Salary: $1250 Salary Raise: $75 New Salary: $1325

Employee #:7902 Current Salary: $3000 Salary Raise: $180 New Salary: $3180

Employee #:7369 Current Salary: $800 Salary Raise: $48 New Salary: $848

Employee #:7788 Current Salary: $3000 Salary Raise: $180 New Salary: $3180

Employee #:7876 Current Salary: $1100 Salary Raise: $66 New Salary: $1166

Total of all raises combined: $1363.5