Examples on Triggers

Instructor: Mohamed Eltabakh
meltabakh@cs.wpi.edu
If the employee salary increased by more than 10%, make sure the ‘rank’ field is not empty and its value has changed, otherwise reject the update.

Create or Replace Trigger EmpSal
Before Update On Employee
For Each Row
Begin
  IF (:new.salary > (:old.salary * 1.1)) Then
    IF (:new.rank is null or :new.rank = :old.rank) Then
      RAISE_APPLICATION_ERROR(-20004, 'rank field not correct');
    End IF;
  End IF;
End;
/

If the trigger exists, then drop it first.

Compare the old and new salaries.

Make sure to have the “/” to run the command.
Example 2

If the employee salary increased by more than 10%, then increment the rank field by 1.

In the case of **Update** event only, we can specify which columns

```sql
Create or Replace Trigger EmpSal
Before Update Of salary On Employee
For Each Row
Begin
  IF (:new.salary > (:old.salary * 1.1)) Then
    :new.rank := :old.rank + 1;
  End IF;
End;
/
```

We changed the new value of **rank** field

The assignment operator has “:=”
Example 3: Using Temp Variable

If the newly inserted record in employee has null hireDate field, fill it in with the current date

Create Trigger EmpDate
Before Insert On Employee
For Each Row
Declare
temp date;
Begin
    Select sysdate into temp from dual;
    IF (:new.hireDate is null) Then
        :new.hireDate := temp;
    End IF;
End;
/

Since we need to change values, then it must be “Before” event

Declare section to define variables

Oracle way to select the current date

Updating the new value of hireDate before inserting it
Example 4: Maintenance of Derived Attributes

Keep the bonus attribute in Employee table always 3% of the salary attribute

Create Trigger *EmpBonus*
Before Insert Or Update On *Employee*
For Each Row
Begin
  :new.bonus := :new.salary * 0.03;
End;
/

The bonus value is always computed automatically

Indicate two events at the same time
Combining Multiple Events in One Trigger

- **If you combine multiple operations**
  - Sometimes you need to know what is the current operation

Create Trigger *EmpBonus*
Before Insert Or Update On *Employee*
For Each Row
Begin
  IF (*inserting*) Then ... End IF;
  IF (*updating*) Then ... End IF;
End;
/

Combine Insert and Update
Can do something different under each operation
Before vs. After

- **Before Event**
  - When checking certain conditions that may cause the operation to be cancelled
    - E.g., if the name is null, do not insert
  - When modifying values before the operation
    - E.g., if the date is null, put the current date

- **After Event**
  - When taking other actions that will not affect the current operations
    - The insert in table X will cause an update in table Y

**Before Insert Trigger:**
```
:new.x := .... //Changing value x that will be inserted
```

**After Insert Trigger:**
```
:new.x := ... //meaningless because the value is already inserted
```
Row-Level vs. Statement-Level Triggers

- **Example**: `Update emp set salary = 1.1 * salary;`
  - Changes many rows (records)

- **Row-level triggers**
  - Check individual values and can update them
  - Have access to `:new` and `:old` vectors

- **Statement-level triggers**
  - Do not have access to `:new` or `:old` vectors (only for row-level)
  - Execute once for the entire statement regardless how many records are affected
  - Used for verification before or after the statement
Example 5: Statement-level Trigger

Store the count of employees having salary > 100,000 in table R

Create Trigger `EmpBonus`
After Insert Or Update of salary Or Delete On `Employee`
For Each Statement
Begin
    delete from R;
    insert into R(cnt) Select count(*) from employee where salary > 100,000;
End;
/

Indicate three events at the same time

Remember: In Oracle, it is not written

Delete the existing record in R, and then insert the new count.
Order Of Trigger Firing

Loop over each affected record

- Before Trigger (statement-level)
- Before Trigger (row-level)
- Event (row-level)
- After Trigger (row-level)
- After Trigger (statement-level)
Some Other Operations

- Dropping Trigger
  
  ```sql
  SQL> Drop Trigger <trigger name>;
  ```

- If creating trigger with errors

  ```sql
  SQL > Show errors;
  ```

  It displays the compilation errors
Example on Triggers

- `branch (branch_name, branch_city, assets)`
- `customer (customer_name, customer_street, customer_city)`
- `account (account_number, branch_name, balance)`
- `loan (loan_number, branch_name, amount)`
- `depositor (customer_name, account_number)`
- `borrower (customer_name, loan_number)`

Sum of loans taken by a customer does not exceed 100,000...
Assume primary keys cannot be updated

- Which table? ➔ Borrower & Loan
- Which event? ➔ Borrower (Insert) & Loan (update)
- Which Timing? ➔ ??? Lets see
- Which Granularity? ➔ row-level
Create Trigger `CustMaxLoan1`

After Insert On `Borrower`

For Each Row

Declare

`sumLoan` int;

Begin

Select sum(amount) into `sumLoan`

From loan L, Borrower B

where L.loan_number = B.loan_number

And B.customer_name = :new.customer_name;

IF `sumLoan` > 100,000 Then

    RAISE_APPLICATION_ERROR(-20004, 'Cannot insert record.');

End IF;

End;
Create Trigger CustMaxLoan2
After Update of amount On Loan
For Each Row
Declare
    sumLoan int; custName varchar2(100);
Begin
    Select customer_name into custName From Borrower
    Where loan_number = :new.loan_number;

    Select sum(amount) into sumLoan
    From loan L, Borrower B
    where L.loan_number = B.loan_number
    And B.customer_name = custName;

    IF sumLoan > 100,000 Then
        RAISE_APPLICATION_ERROR(-20004, 'Cannot insert record.');
    End IF;
End;
/

Sum of loans taken by a customer does not exceed 100,000
Example 2

- branch (branch_name, branch_city, assets)
- customer (customer_name, customer_street, customer_city)
- account (account_number, branch_name, balance)
- loan (loan_number, branch_name, amount)
- depositor (customer_name, account_number)
- borrower (customer_name, loan_number)

Sum of loans taken by a customer does not exceed 100,000… Assume primary keys cannot be updated

For each table, create “Before Update” trigger preventing a change on PK columns

Create Trigger PK-No-Update
Before Update of loan_number On Loan
For Each Row
Begin
    RAISE_APPLICATION_ERROR(-20004, 'Cannot Update PK...');
End;
/

What if you are requested to enforce this part ???