# OPIS TRIGGERÓW ORAZ KOD JE GENERUJĄCY

Przypomnienie ról systemowych:

* Chef - odczyt OrderDetails
* Accountant - generowanie faktur i raportów, przeglądanie Orders, kursy walut
* Waiter - dostep do Orders i OrdersDetails
* Menager - nadzoruje całość, kontrola Employees
* Administrator IT - kontrola nad całym systemem
* WarehouseMenager - nadzoruje zamówienia i dba o uzupełnianie warehouse

Każdy trigger zawiera:

1. krótki opis
2. warunek aktywacji
3. kod triggera.

### Ustawienie unitPrice

* Automatycznie wstawia cenę jednostkową z Warehouse do OrderDetails
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgOrderDetailPriceSync  
 ON OrderDetails  
 AFTER INSERT AS  
BEGIN  
 UPDATE OrderDetails  
 SET OrderDetails.UnitPrice = Warehouse.UnitPrice  
 FROM OrderDetails  
 INNER JOIN Warehouse on Warehouse.MealID = OrderDetails.MealID  
 INNER JOIN Orders on Orders.OrderID = OrderDetails.OrderID  
 WHERE OrderDetails.MealID = Warehouse.MealId  
 AND OrderDetails.UnitPrice <> Warehouse.UnitPrice  
 AND Orders.OrderID = (SELECT TOP 1 OrderDetails.OrderID FROM OrderDetails ORDER BY OrderID DESC)  
END  
GO;

### Czy rezerwacja jest możliwa

* Weryfikuje czy klient spełnia warunki wymagane do złożenia rezerwacji, jeśli nie to usuwa zamówienie
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgCanReservate  
 ON OrderDetails  
 AFTER INSERT AS  
BEGIN  
 DECLARE @latestOrderID INT  
 SET @latestOrderID = (SELECT TOP 1 OrderID  
 FROM OrderDetails  
 ORDER BY OrderID DESC)  
 DECLARE @ifReservation bit  
 SET @ifReservation = IIF((SELECT (CAST(O.WhenOrdered AS INT) - CAST(O.WhenToBeDone AS INT)) \* O.SeatsNeeded  
 FROM Orders O  
 INNER JOIN OrderDetails OD on OD.OrderID = O.OrderID  
 WHERE OD.OrderID = @latestOrderID) < 0, 1, 0)  
 DECLARE @customerID int  
 SET @customerID = (  
 SELECT O.CustomerID  
 FROM Orders O  
 INNER JOIN OrderDetails OD on OD.OrderID = O.OrderID  
 WHERE OD.OrderID = @latestOrderID)  
 DECLARE @ifIndividual bit  
 SET @ifIndividual = IIF(@customerID is not NULL, 1, 0)  
  
 DECLARE @howManyTimes INT;  
 EXECUTE uspHowManyVisits  
 @clientID = @customerID,  
 @howManyVisits = @howManyTimes OUTPUT;  
  
 DECLARE @totalCost int  
 SET @totalCost = (  
 SELECT SUM(Quantity \* W.UnitPrice) AS sum  
 FROM OrderDetails OD  
 INNER JOIN Orders O on O.OrderID = OD.OrderID  
 INNER JOIN Warehouse W on W.MealID = OD.MealID  
 WHERE O.OrderID = @latestOrderID  
 GROUP BY WhenOrdered)  
 DECLARE @WZ DECIMAL(6, 2)  
 DECLARE @WK DECIMAL(6, 2)  
 SET @WZ = (SELECT VarValue FROM Variables WHERE VarName LIKE 'WZ')  
 SET @WK = (SELECT VarValue FROM Variables WHERE VarName LIKE 'WK')  
  
 DECLARE @conditions bit  
 SET @conditions = IIF(@howManyTimes >= @WK AND @totalCost >= @WZ, 1, 0)  
 IF @ifReservation = 1 AND @ifIndividual = 1 AND @conditions = 1  
 BEGIN  
 PRINT ('reserved')  
 END  
 ELSE  
 BEGIN  
 PRINT '(Can not make reservation)'  
 DELETE FROM Orders WHERE OrderID = @latestOrderID  
 END  
END  
GO;

### Czy jest wystarczająco miejsc

* Weryfikuje czy restauracja posiada wystarczająco wolnych siedzeń dla obecnego zamówienia, jeśli nie to usuwa zamówienie
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgEnoughSeats  
 ON Orders  
 AFTER INSERT AS  
BEGIN  
 DECLARE @LatestOrderID INT  
 SET @LatestOrderID = (SELECT TOP 1 OrderID  
 FROM Orders  
 ORDER BY OrderID DESC)  
 DECLARE @orderWhenToBeDone datetime  
 SET @orderWhenToBeDone = (SELECT WhenToBeDone  
 FROM Orders  
 WHERE Orders.OrderID = @LatestOrderID)  
 DECLARE @orderForHowLong time  
 SET @orderForHowLong = (SELECT ForHowLong  
 FROM Orders  
 WHERE Orders.OrderID = @LatestOrderID)  
 DECLARE @freeSeats INT  
 SET @freeSeats = (SELECT SUM(T.SeatNo)  
 FROM Tables T  
 INNER JOIN TablesUsed TU on TU.TableID = T.TableID  
 INNER JOIN Orders O on O.OrderID = TU.OrderID  
 WHERE O.WhenToBeDone > @orderWhenToBeDone + CAST(@orderForHowLong AS DATETIME)  
 OR O.WhenToBeDone + CAST(O.ForHowLong AS DATETIME) < @orderWhenToBeDone)  
 DECLARE @orderSeatsNeeded INT  
 SET @orderSeatsNeeded = (SELECT O.SeatsNeeded  
 FROM Orders O  
 WHERE O.OrderID = @LatestOrderID)  
 IF @orderSeatsNeeded > @freeSeats  
 BEGIN  
 PRINT ('no enought free tables')  
 DELETE FROM Orders WHERE OrderID = @LatestOrderID  
 END  
END  
GO;

### Czy zamówiono owoce morza z wyprzedzeniem

* Weryfikuje czy zamówienie na owoce morza zostało złożone z wystarczającym wyprzedzeniem, jeśli to nie usuwa zamówienie
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgSeafoodEarlyEnough  
 ON OrderDetails  
 AFTER INSERT AS  
BEGIN  
 DECLARE @LatestOrderID INT  
 SET @LatestOrderID = (SELECT TOP 1 OrderID  
 FROM OrderDetails  
 ORDER BY OrderID DESC)  
 DECLARE @dayOfOrder INT  
 SET @dayOfOrder = (SELECT DAY(O.WhenOrdered)  
 FROM Orders O  
 WHERE O.OrderID = @LatestOrderID)  
 DECLARE @dayOfDone INT  
 SET @dayOfOrder = (SELECT DAY(O.WhenToBeDone)  
 FROM Orders O  
 WHERE O.OrderID = @LatestOrderID)  
 DECLARE @weekdayOfDone INT  
 SET @weekdayOfDone = (SELECT DATEPART(weekday, O.WhenToBeDone)  
 FROM Orders O  
 WHERE O.OrderID = @LatestOrderID)  
 DECLARE @ifSeafood bit  
 SET @ifSeafood = IIF('Seafood' IN (SELECT CategoryName  
 FROM Categories C  
 INNER JOIN Warehouse W on W.CategoryID = C.CategoryID  
 INNER JOIN OrderDetails OD on OD.MealID = W.MealID  
 WHERE OD.OrderID = @LatestOrderID), 1, 0)  
 DECLARE @ifEarly bit  
 SET @ifEarly = IIF(@dayOfOrder < (@dayOfOrder - (@weekdayOfDone - 2)), 1,0)  
 IF @ifSeafood = 1 AND @ifEarly = 0  
 BEGIN  
 PRINT ('you need to order seafood earlier')  
DELETE FROM Orders WHERE OrderID = @LatestOrderID  
 END  
END  
GO;

### Czy zamówiono owoce morza na odpowiedni dzień

* Weryfikuje czy zamówienie na owoce morza zostało złożone na odpowiedni dzień (czwartek/piątek/sobota), jeśli to nie usuwa zamówienie
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgSeafoodForDay  
 ON OrderDetails  
 AFTER INSERT AS  
BEGIN  
 DECLARE @LatestOrderID INT  
 SET @LatestOrderID = (SELECT TOP 1 OrderID  
 FROM OrderDetails  
 ORDER BY OrderID DESC)  
 DECLARE @ifSeafood bit  
 SET @ifSeafood = IIF('Seafood' IN (SELECT CategoryName  
 FROM Categories C  
 INNER JOIN Warehouse W on W.CategoryID = C.CategoryID  
 INNER JOIN OrderDetails OD on OD.MealID = W.MealID  
 WHERE OD.OrderID = @LatestOrderID), 1 , 0)  
 DECLARE @weekdayOfDone INT  
 SET @weekdayOfDone = (SELECT DATEPART(weekday, O.WhenToBeDone)FROM Orders O  
 WHERE O.OrderID = @LatestOrderID)  
 DECLARE @ifEndOfWeek bit  
 SET @ifEndOfWeek = IIF(@weekdayOfDone <= 7 AND @weekdayOfDone >= 5, 1, 0)  
 IF @ifSeafood = 1 AND @ifEndOfWeek = 0  
 BEGIN  
 PRINT ('you can order seafood only for czw/pt/sb')  
DELETE FROM Orders WHERE OrderID = @LatestOrderID  
 END  
END  
GO;

### Zniżka wieczysta

* Jeśli klient składający zamówienie nie ma jeszcze zniżki wieczystej a spełnia warunki to zostaje ona dodana
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgRegularCustomerDiscount  
 ON OrderDetails  
 AFTER INSERT AS  
BEGIN  
 DECLARE @LatestOrderID INT  
 SET @LatestOrderID = (SELECT TOP 1 OrderID  
 FROM OrderDetails  
 ORDER BY OrderID DESC)  
 DECLARE @CustomerID INT  
 SET @CustomerID = (SELECT CustomerID FROM Orders WHERE Orders.OrderID = @LatestOrderID)  
 IF @CustomerID IS NOT NULL  
 BEGIN  
 DECLARE @DiscountExists INT  
 SET @DiscountExists = (SELECT TOP 1 DiscountID  
 FROM Discounts  
 WHERE Discounts.CustomerID = @CustomerID  
 AND DiscountType LIKE 'Loyal Customers discount')  
 IF @DiscountExists IS NULL  
 BEGIN  
 DECLARE @Z1 INT, @K1 INT  
 SET @Z1 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'Z1')  
 SET @K1 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'K1')  
 IF (SELECT COUNT(\*)  
 FROM Orders  
 WHERE CustomerID = @CustomerID  
 AND (SELECT SUM(Quantity \* W.UnitPrice) as sum  
 FROM OrderDetails OD  
 INNER JOIN Orders O on O.OrderID = OD.OrderID  
 INNER JOIN Warehouse W on W.MealID = OD.MealID  
 WHERE O.OrderID = @LatestOrderID) >= @K1) >= @Z1  
 BEGIN  
 DECLARE @R1 FLOAT  
 SET @R1 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'R1')  
 INSERT INTO master.dbo.Discounts (CustomerID, DiscountType, DiscountValue)  
 VALUES (@CustomerID, N'Loyal Customers discount', @R1)  
 END  
 END  
 END  
 ELSE  
 BEGIN  
 DECLARE @tmp INT  
 END  
  
END  
GO;

### Zniżka jednorazowa

* Jeśli klient składający zamówienie nie ma aktualnie zniżki jednorazowej a spełnia warunki to zostaje ona dodana
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgSingleUseDiscount  
 ON OrderDetails  
 AFTER INSERT AS  
BEGIN  
 DECLARE @LatestOrderID INT  
 SET @LatestOrderID = (SELECT TOP 1 OrderID  
 FROM OrderDetails  
 ORDER BY OrderID DESC)  
 DECLARE @CustomerID INT  
 SET @CustomerID = (SELECT DISTINCT CustomerID FROM Orders WHERE Orders.OrderID = @LatestOrderID)  
 IF @CustomerID IS NOT NULL  
 BEGIN  
 DECLARE @SingleDiscountExists INT, @K2 INT, @Spendings INT  
 DECLARE @R2 DECIMAL(6, 2)  
 SET @SingleDiscountExists = (SELECT TOP 1 DiscountID  
 FROM Discounts  
 WHERE CustomerID = @CustomerID  
 AND DiscountType LIKE 'One use discount')  
 SET @K2 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'K2')  
 SET @R2 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'R2')  
 SET @Spendings = 0  
 IF @SingleDiscountExists IS NULL  
 BEGIN  
 SET @Spendings = (SELECT SUM(Quantity \* W.UnitPrice) as sum  
 FROM OrderDetails OD  
 INNER JOIN Orders O on O.OrderID = OD.OrderID  
 INNER JOIN Warehouse W on W.MealID = OD.MealID  
 WHERE O.OrderID = @LatestOrderID)  
 IF @Spendings >= @K2  
 BEGIN  
 INSERT INTO master.dbo.Discounts(CustomerID, DiscountType, DiscountValue)  
 VALUES (@CustomerID, 'One use discount', @R2)  
 END  
 END  
 ELSE  
 BEGIN  
 DECLARE @WhenWasIt DATE, @D1 INT  
 SET @D1 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'D1')  
 SET @WhenWasIt = (SELECT TOP 1 Until  
 FROM Discounts  
 WHERE CustomerID = 1  
 AND DiscountType LIKE 'One use discount'  
 ORDER BY Until DESC)  
 SET @Spendings = (SELECT SUM(Quantity \* W.UnitPrice) as sum  
 FROM OrderDetails OD  
 INNER JOIN Orders O on O.OrderID = OD.OrderID  
 INNER JOIN Warehouse W on W.MealID = OD.MealID  
 WHERE O.OrderID = @LatestOrderID  
 AND O.WhenOrdered > @WhenWasIt)  
 IF @Spendings >= @K2  
 BEGIN  
 INSERT INTO master.dbo.Discounts(CustomerID, DiscountType, DiscountValue)  
 VALUES (@CustomerID, 'One use discount', @R2)  
 END  
 END  
 END  
END  
GO;

### Ważność zniżki

* Ustawia datę ważności dodawanej zniżki
* Warunek aktywacji: po dodaniu do Discounts

CREATE TRIGGER trgSetUntil  
 ON Discounts  
 AFTER INSERT AS  
BEGIN  
 DECLARE @D1 INT, @LatestID INT  
 SET @D1 = (SELECT VarValue FROM Variables WHERE VarName LIKE 'D1')  
 SET @LatestID = (SELECT TOP 1 DiscountID FROM Discounts ORDER BY DiscountID DESC)  
 UPDATE master.dbo.Discounts SET Until = GETDATE() + @D1 WHERE DiscountID = @LatestID  
END  
GO;

### Zużycie zniżki

* Jeśli klient używa zniżki, to jest ona zaznaczana jako wykorzystana
* Warunek aktywacji: po dodaniu nowego zamówienia

CREATE TRIGGER trgCheckDiscountUse  
 ON Orders  
 AFTER INSERT AS  
BEGIN  
 DECLARE @LatestID INT, @DiscountID INT  
 SET @LatestID = (SELECT TOP 1 OrderID FROM Orders ORDER BY OrderID DESC)  
 SET @DiscountID = (SELECT Orders.DiscountID FROM Orders WHERE OrderID = @LatestID)  
 IF @DiscountID IS NOT NULL  
 BEGIN  
 UPDATE master.dbo.Discounts SET IfUsed = 1 WHERE DiscountID = @DiscountID  
 END  
END  
GO;