Triggery

Add_student_to_modules_trigger - PS

Trigger ten odpowiada za przypisanie użytkownika do modułów danego kursu, którego zakupił. Wartości te są wpisywane do tablicy "Users_Modules_Passes" z domyślną wartością "null". Trigger reaguje w sytuacji gdy przypiszemy użytkownika do kursu, czyli pojawienie się wpisu w "Users_Courses".

```
create trigger add_student_to_modules_trigger
    on Users_Courses
    after insert
as begin
    begin try
        declare @UserID int, @CourseID int;

        -- Pobranie informacji o kursie i użytkownikowi po kupieniu
        select @UserID = UserID, @CourseID = CourseID
        from inserted;

        -- Pobrać tabele z modułami danego kursu
        declare @CourseModuleID table (
            ModuleID int
        );

        -- Zebranie ID modułów danego kursu
        insert @CourseModuleID (ModuleID)
        select ModuleID from Modules where Modules.CourseID = @CourseID;

        -- Dodanie wartości użytkownika do danego modułu
        insert Users_Modules_Passes (UserID, ModuleID, Passed)
        select @UserID, ModuleID, null
        from @CourseModuleID;
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end
```

Add_student_to_product_trigger - PS

Trigger ten odpowiada za dodanie użytkownika do odpowiedniego produktu w zależności co użytkownik zakupił. Po pojawieniu się zamówienia w "Orders_Details" mamy informacje o użytkwniku i co on zakupił. Następnie w zależności co kupił wpisywany jest do tabeli z odpowiednią kategorią produktu.

```
create trigger add_student_to_product_trigger
    on Orders_Details
    after insert
as begin
    begin try
        set nocount on; -- dla poprawy wydajności

        declare @CategoryName nvarchar(15);
        declare @ProductID int;

        -- Wyciągnieci ID produktu
        select @ProductID = ProductID from inserted;

        -- Znalezienie nazwy kategorii kupionego produktu
        select @CategoryName = Name from Products inner join Categories on Products.CategoryID = Categories.CategoryID
                                                    where Products.ProductID = @ProductID;

        -- Produkt -> webinar
        if @CategoryName = 'Webinar'
        begin
            -- Sprawdzenie czy już jest zapisany na dany webinar
            if exists(select UserID from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
                                    inner join Orders on Orders_Details.OrderID = Orders.OrderID
                                    where UserID in (select distinct UserID from Users_Webinars inner join inserted on inserted.ProductID =
Users_Webinars.WebinarID))
            begin
                throw 50001, 'Użytkownik o podanym ID jest już zapisany na ten webinar', 1;
            end

            -- W innym przypadku dodajemy go do webinaru
            insert Users_Webinars (UserID, WebinarID)
            select UserID, inserted.ProductID
            from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
            inner join Orders on Orders_Details.OrderID = Orders.OrderID
        end
```

```
        -- Produkt -> kurs
        if @CategoryName = 'Course'
        begin
            -- Sprawdzenie czy już jest zapisany na dany kurs
            if exists(select UserID from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
                                    inner join Orders on Orders_Details.OrderID = Orders.OrderID
                                    where UserID in (select distinct UserID from Users_Courses inner join inserted on inserted.ProductID =
Users_Courses.CourseID))
            begin
                throw 50002, 'Użytkownik o podanym ID jest już zapisany na ten kurs', 1;
            end

            -- W innym przypadku dodajemy go do webinaru
            insert Users_Courses (UserID, CourseID)
            select UserID, inserted.ProductID
            from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
            inner join Orders on Orders_Details.OrderID = Orders.OrderID
        end

        -- Produkt -> studia
        if @CategoryName = 'Studies'
        begin
            -- Sprawdzenie czy już jest zapisany na dane studia
            if exists(select UserID from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
                                    inner join Orders on Orders_Details.OrderID = Orders.OrderID
                                    where UserID in (select distinct UserID from Users_Studies inner join inserted on inserted.ProductID =
Users_Studies.StudiesID))
            begin
                throw 50003, 'Użytkownik o podanym ID jest już zapisany na te studia', 1;
            end

            -- W innym przypadku dodajemy go do webinaru
            insert Users_Studies (UserID, StudiesID, Grade)
            select UserID, inserted.ProductID, null
            from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
            inner join Orders on Orders_Details.OrderID = Orders.OrderID
        end

        -- Produkt -> spotkanie studyjne
        if @CategoryName = 'Meeting'
        begin
            -- Sprawdzenie czy już jest zapisany na dane spotkanie studyjne
            if exists(select UserID from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
                                    inner join Orders on Orders_Details.OrderID = Orders.OrderID
                                    where UserID in (select distinct UserID from Users_Meetings_Attendance inner join inserted on
inserted.ProductID = Users_Meetings_Attendance.MeetingID))
            begin
                throw 50004, 'Użytkownik o podanym ID jest już zapisany na dane spotkanie studyjne', 1;
            end

            -- W innym przypadku dodajemy go do webinaru
            insert Users_Meetings_Attendance (UserID, MeetingID, Present)
            select UserID, inserted.ProductID, null
            from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
            inner join Orders on Orders_Details.OrderID = Orders.OrderID
        end
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;
```

Add_meetings_to_attendance_trigger - MS

Trigger ten dodaje użytkowników, którzy zakupili studia do spotkań studyjnych bez wpsianej narazie obecności. Po pojawieniu się wpisu w "Users_Studies" trigger pobiera wszystkie spotkania studyjne danych studiów i przypisuje użytkownika do każdego spotkania dodając go do "Users_Meetings_Attendance".

```
create trigger add_meetings_to_attendance_trigger
    on Users_Studies
    after insert
as begin
    begin try
        declare @UserID int, @StudiesID int;

        -- Pobranie informacji o studiach i użytkowniku po kupieniu
        select @UserID = UserID, @StudiesID = StudiesID
        from inserted;

        -- Stworzenie tabeli ze wszystkimi spotkaniami ze wszystkich zjazdów z danych studiów
```

```
        declare @StudiesReunionsMeetingsIDs table (
            MeetingID int
        );

        -- Zebranie ID spotkań ze wszystkich zjazdów z danych studiów
        insert @StudiesReunionsMeetingsIDs
        select m.MeetingID from Studies s
            join Studies_Reunion r on s.StudiesID = r.StudiesID
            join Meetings m on r.ReunionID = m.ReunionID
            where s.StudiesID = @StudiesID;

        -- Dodanie pola przechowującego informację o obecności użytkownika na danym spotkaniu
        insert Users_Meetings_Attendance (UserID, MeetingID, Present)
        select @UserID, MeetingID, null
        from @StudiesReunionsMeetingsIDs;
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end
```

Add_reunions_to_suborders_trigger - MS

Celem triggera jest przypisanie użytkownika do zjazdów studyjnych studiów, które zakupił. Zostaje on dopisany do tablicy "Payment_for_reunions" ze wszystkimi zjazdami z danych studiów. Tablica ta sama będzie przechowywała informacje o tym czy użytkownik zapłacił za dany zjazd.

```
create trigger add_reunions_to_suborders_trigger
    on Orders_Details
    after insert
as begin
    begin try
        set nocount on; -- dla poprawy wydajności

        declare @CategoryName nvarchar(15);
        declare @ProductID int;
        declare @SubOrderID int;

        -- Wyciągniecie ID produktu
        select @ProductID = ProductID, @SubOrderID = SubOrderID from inserted;

        -- Znalezienie nazwy kategorii kupionego produktu
        select @CategoryName = Name from Products inner join Categories on Products.CategoryID = Categories.CategoryID
                                                    where Products.ProductID = @ProductID;

        if @CategoryName = 'Studies'
        begin
            -- Sprawdzenie, czy użytkownik już jest zapisany na dane studia
            if not exists(select UserID from inserted inner join Orders_Details on inserted.SubOrderID = Orders_Details.SubOrderID
                                    inner join Orders on Orders_Details.OrderID = Orders.OrderID
                                    where UserID in (select distinct UserID from Users_Studies inner join inserted on inserted.ProductID =
Users_Studies.StudiesID))
            begin
                throw 50001, 'Użytkownik nie został dodany do studiów', 1;
            end

            -- W przeciwnym wypadku dodajemy do Orders_Details podzamówienia ze wszystkimi zjazdami z danych studiów
            declare @allReunionsForStudies table (
                    PaymentDeadline date,
                    FullPrice money,
                    ReunionID int
            )

            insert @allReunionsForStudies (PaymentDeadline, FullPrice, ReunionID)
            select dateadd(day, -3, r.StartDate), r.Price, r.ReunionID
                from Studies_Reunion r
                where StudiesID = @ProductID

            insert Payment_for_reunions (SubOrderID, ReunionID, PaymentDeadline, PaymentDate, IsPaid)
            select @SubOrderID, ReunionID, PaymentDeadline, null, null
            from @allReunionsForStudies;
        end
    end try
    begin catch
        -- Przerzucenie ERRORa dalej
        throw;
    end catch
end;
```