Funkcje

Check_translator_language

Funkcja dostaje parę indeksów, język i tłumacz, a następnie sprawdza czy dany tłumacz zna podany język.

```
create function check_translator_language(
    @TranslatorID int,
    @LanguageID int
)
returns bit
as begin
    -- 1 - para istnieje, 0 - para nie istnieje
    declare @Result bit;

    -- Sprawdzenie czy dana para istnieje
    if exists(select 1 from Translators_Languages where TranslatorID = @TranslatorID and
                                                        LanguageID = @LanguageID)
    begin
        set @Result = 1;
    end

    -- Domyślnie zakładamy, że jak nie ma tłumacza to język czegoś to Polski
    else if @TranslatorID is null and @LanguageID = 19
    begin
        set @Result = 1;
    end
    else
    begin
        set @Result = 0;
    end

    return @Result;
end
```

Check_translator_availability

Funkcja dostaje ID tłumacza, datę i czas rozpoczęcia zajęć oraz czas trawania zajęć. Jej celem jest sprawdzenie czy podany tłumacz ma inne zajęcia w tym czasie. Sprawdzane są wszystkie jego możliwe aktywności (webinary, spotkania studyjne, moduły).

```
create function check_translator_availability(
    @TranslatorID int,
    @DateAndBeginningTime datetime,
    @Duration time(0)
) returns bit
as begin
    -- 1 - jakieś spotkania nakładają się, 0 - nic się nei nakłada
    declare @Result bit = 0;

    declare @StartDate datetime = @DateAndBeginningTime;
    declare @EndDate datetime = dateadd(minute, datediff(minute, 0, @Duration), @DateAndBeginningTime);

    -- Zadekalrowanie tabeli ze wszystkimi spotkaniami tłumacza
    declare @TranslatorActivities table (
        DateAndBeginningTime datetime,
        Duration time(0)
                                        );

    -- Moduły stacjonarne
    insert @TranslatorActivities
    select DateAndBeginningTime, Duration
    from In_person_Modules inner join Modules on In_person_Modules.ModuleID = Modules.ModuleID
    where TranslatorID = @TranslatorID;

    -- Moduły synchroniczne
    insert @TranslatorActivities
    select DateAndBeginningTime, Duration
    from Online_Sync_Modules inner join Modules on Online_Sync_Modules.ModuleID = Modules.ModuleID
    where TranslatorID = @TranslatorID;

    -- Spotkania stacjonarne
    insert @TranslatorActivities
    select DateAndBeginningTime, Duration
    from In_person_Meetings inner join Meetings on In_person_Meetings.MeetingID = Meetings.MeetingID
    where TranslatorID = @TranslatorID;
```

```
    -- Spotkania synchronoczne
    insert @TranslatorActivities
    select DateAndBeginningTime, Duration
    from Online_Sync_Meetings inner join Meetings on Online_Sync_Meetings.MeetingID = Meetings.MeetingID
    where TranslatorID = @TranslatorID;

    -- Sprawdzenie czy date się nie nakładają
    if exists(select 1 from @TranslatorActivities where (
        @StartDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) or
        @EndDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) or
        (@StartDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) and
         @EndDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime)) or
        (DateAndBeginningTime between @StartDate and @EndDate and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) between
@StartDate and @EndDate)
        ))
    begin
        set @Result = 1;
    end

    return @Result;
end
```

Check_teachers_availability

Funkcja otrzymuje ID nauczyciela, datę i czas rozpoczęcia zajęć oraz czas ich trwania. Jej celem jest sprawdzenei czy dany nauczyciel nie ma w tym czasie innych aktywności (spotkania studyjne, webinary, moduły).

```
create function check_teachers_availability(
    @TeacherID int,
    @DateAndBeginningTime datetime,
    @Duration time(0)
) returns bit
as begin
    -- 1 - nauczyciel zajęty w danym terminie, 0 - nauczyciel wolny w danym terminie
    declare @Result bit = 0;

    declare @StartDate datetime = @DateAndBeginningTime
    declare @EndDate datetime = dateadd(minute, datediff(minute, 0, @Duration), @DateAndBeginningTime);

    -- Zadeklarowanie tabeli ze wszystkimi spotkaniami nauczyciela
    declare @TeacherActivities table (
        DateAndBeginningTime datetime,
        Duration time(0)
                                    );

    -- Moduły
    insert @TeacherActivities
    select DateAndBeginningTime, Duration
    from Modules where TeacherID = @TeacherID

    -- Spotkania studyjne
    insert @TeacherActivities
    select DateAndBeginningTime, Duration
    from Meetings where TeacherID = @TeacherID

    -- Webinary
    insert @TeacherActivities
    select DateAndBeginningTime, Duration
    from Webinars where TeacherID = @TeacherID

    -- Sprawdzenie czy date się nie nakładają
    if exists(select 1 from @TeacherActivities where (
        @StartDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) or
        @EndDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) or
        (@StartDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) and
         @EndDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime)) or
        (DateAndBeginningTime between @StartDate and @EndDate and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) between
@StartDate and @EndDate)
        ))
    begin
        set @Result = 1;
    end

    return @Result;
end
```

Check_classroom_availability

Funkcja dostaje numer klasy oraz ID modułu lub spotkania. Jej celem jest sprawdzenie czy podany moduł lub spotkanie może się odbyć w tej sali na podstawie czasu rozpoczęcia danego modułu lub spotkania.

```
create function check_classroom_availability(
    @Classroom int,
    @DateAndBeginningTime datetime,
    @Duration time(0)
) returns bit
as begin
    -- 1 - sala zajęta, 0 - sala wolna
    declare @Result bit = 0;

    declare @StartDate datetime = @DateAndBeginningTime
    declare @EndDate datetime = dateadd(minute, datediff(minute, 0, @Duration), @DateAndBeginningTime);

    -- 1. Pobranie danych o czasie danego modułu dla danej sali
    declare @ClassroomUsage table (
        DateAndBeginningTime datetime,
        Duration time(0)
                                )

    -- Moduły stacjonarne
    insert @ClassroomUsage
    select DateAndBeginningTime, Duration
    from In_person_Modules inner join Modules on In_person_Modules.ModuleID = Modules.ModuleID
    where Classroom = @Classroom

    -- Spotkania stacjonarne
    insert @ClassroomUsage
    select  DateAndBeginningTime, Duration
    from In_person_Meetings inner join Meetings on In_person_Meetings.MeetingID = Meetings.MeetingID
    where Classroom = @Classroom

    -- Sprawdzenie czy w danym okresie nie ma żadnego spotkania studyjnego w danej sali
    if exists(select 1 from @ClassroomUsage where (
        @StartDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) or
        @EndDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) or
        (@StartDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) and
         @EndDate between DateAndBeginningTime and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime)) or
        (DateAndBeginningTime between @StartDate and @EndDate and dateadd(minute, datediff(minute, 0, Duration), DateAndBeginningTime) between
@StartDate and @EndDate)
        ))
    begin
        set @Result = 1;
    end

    return @Result;
end
```

Check_product_availability

Funkcja dostaje ID danego produktu i sprawdza czy jest on dostępny dla użytkowników, czyli czy STATUS jest równy 1.

```
create function check_product_availability(
    @ProductID int
) returns bit
as begin
    -- 1 - produkt dostępy, 0 - produkt nie dostępny
    declare @Result bit = 0;

    -- Sprawdzenie statusu
    if (select Status from Products where ProductID = @ProductID) = 1
    begin
        set @Result = 1;
    end

    return @Result
end
```

Check_user_enrollment_for_product

Funkcja dostaje ID użytkownika i ID produktu (studiów/kursu/webinaru) i sprawdza, czy użytkownik zapisał się do danych studiów/kursu/webinaru

```
create function check_user_enrollment_for_product(
    @UserID int,
    @ProductID int
```

```
)
returns bit
as begin
    -- 1 - para istnieje, 0 - para nie istnieje
    declare @Result bit;

    -- Sprawdzenie czy dana para istnieje
    if exists(select 1 from Users_Studies where UserID = @UserID and
                                                StudiesID = @ProductID
       union select 1 from Users_Courses where UserID = @UserID and
                                                CourseID = @ProductID
       union select 1 from Users_Webinars where UserID = @UserID and
                                                WebinarID = @ProductID)
    begin
        set @Result = 1;
    end

    else
    begin
        set @Result = 0;
    end

    return @Result;
end
```