# --create/alter tenancytenant trigger

CREATE OR ALTER TRIGGER [dbo].[trg\_TenancyTenant\_History]

ON [dbo].[TenancyTenant]

AFTER INSERT, DELETE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @UserID INT;

SELECT @UserID = UserID

FROM ApplicationUser

WHERE ActiveUser = 1;

-- Log entries for newly added tenants

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, UserID, OrganizationID)

SELECT

GETDATE() AS ChangeDate,

'INSERT' AS ChangeOperation,

'Tenant' AS FieldChanged,

NULL AS OldValue,

CAST(i.TenantID AS NVARCHAR) AS NewValue,

i.TenancyID,

@UserID AS UserID,

o.OrganizationID

FROM

inserted i

INNER JOIN

Tenancy t ON i.TenancyID = t.TenancyID

INNER JOIN

Organization o ON t.OrganizationID = o.OrganizationID;

-- Log entries for removed tenants

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, UserID, OrganizationID)

SELECT

GETDATE() AS ChangeDate,

'DELETE' AS ChangeOperation,

'Tenant' AS FieldChanged,

CAST(d.TenantID AS NVARCHAR) AS OldValue,

NULL AS NewValue,

d.TenancyID,

@UserID AS UserID,

o.OrganizationID

FROM

deleted d

INNER JOIN

Tenancy t ON d.TenancyID = t.TenancyID

INNER JOIN

Organization o ON t.OrganizationID = o.OrganizationID;

SET NOCOUNT OFF;

END;

GO

ALTER TABLE [dbo].[TenancyTenant] ENABLE TRIGGER [trg\_TenancyTenant\_History]

GO

# --create/alter tenancy trigger

CREATE OR ALTER TRIGGER [dbo].[trg\_Tenancy\_History]

ON [dbo].[Tenancy]

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @TenancyID INT;

DECLARE @UserID INT;

SELECT @UserID = UserID

FROM ApplicationUser

WHERE ActiveUser = 1;

-- Handle INSERT operations

IF EXISTS (SELECT 1 FROM inserted) AND NOT EXISTS (SELECT 1 FROM deleted)

BEGIN

DECLARE InsertedRows CURSOR FOR

SELECT TenancyID FROM inserted;

OPEN InsertedRows;

FETCH NEXT FROM InsertedRows INTO @TenancyID;

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'INSERT', NULL, NULL, NULL, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

FETCH NEXT FROM InsertedRows INTO @TenancyID;

END;

CLOSE InsertedRows;

DEALLOCATE InsertedRows;

END;

-- Handle UPDATE operations

IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)

BEGIN

DECLARE @OldValue NVARCHAR(MAX);

DECLARE @NewValue NVARCHAR(MAX);

DECLARE UpdatedRows CURSOR FOR

SELECT TenancyID FROM inserted;

OPEN UpdatedRows;

FETCH NEXT FROM UpdatedRows INTO @TenancyID;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Log changes for TenancyStatus

SET @OldValue = (SELECT TenancyStatus FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT TenancyStatus FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'TenancyStatus', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for IsDeleted (soft delete)

SET @OldValue = (SELECT IsDeleted FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT IsDeleted FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'IsDeleted', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END;

-- Log changes for MoveInDate

SET @OldValue = (SELECT MoveInDate FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT MoveInDate FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'MoveInDate', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for MoveOutDate

SET @OldValue = (SELECT MoveOutDate FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT MoveOutDate FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'MoveOutDate', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for SquareMeter

SET @OldValue = (SELECT SquareMeter FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT SquareMeter FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'SquareMeter', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for Rent

SET @OldValue = (SELECT Rent FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT Rent FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'Rent', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for Rooms

SET @OldValue = (SELECT Rooms FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT Rooms FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'Rooms', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for BathRooms

SET @OldValue = (SELECT BathRooms FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT BathRooms FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'BathRooms', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for PetsAllowed

SET @OldValue = (SELECT PetsAllowed FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT PetsAllowed FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'PetsAllowed', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for AddressID

SET @OldValue = (SELECT AddressID FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT AddressID FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'AddressID', @OldValue, @NewValue, @TenancyID, (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID), @UserID);

END

-- Log changes for OrganizationID

SET @OldValue = (SELECT OrganizationID FROM deleted WHERE TenancyID = @TenancyID);

SET @NewValue = (SELECT OrganizationID FROM inserted WHERE TenancyID = @TenancyID);

IF @OldValue <> @NewValue

BEGIN

INSERT INTO History (ChangeDate, ChangeOperation, FieldChanged, OldValue, NewValue, TenancyID, OrganizationID, UserID)

VALUES (GETDATE(), 'UPDATE', 'OrganizationID', @OldValue, @NewValue, @TenancyID, @NewValue, @UserID);

END

FETCH NEXT FROM UpdatedRows INTO @TenancyID;

END;

CLOSE UpdatedRows;

DEALLOCATE UpdatedRows;

END;

end;

GO

ALTER TABLE [dbo].[Tenancy] ENABLE TRIGGER [trg\_Tenancy\_History]

GO