Test av Lagrade Procedurer

Testmål

Målet är att kontrollera mina Lagrade Procedurer utifrån den funktion de ska ha. Jag kommer använda SQL server management studio som test miljö.

Följande Lagrade Procedurer testas

Namn	Användningsområde
usp_MedlemsLista	Visar lista på alla medlemmar och sorterar på efternamnet
usp_GetMedlem	Visar detaljerad information om enskild medlem
usp_AddMedlem	lägger till medlem
usp_DeleteMedlem	Tar bort vald medlem
usp_UpdateMedlem	Uppdaterar vald medlem
usp_GetMedlemKontaktByID	Visar specifik medlems kontakt information
usp_AllaKontakttyper	visar allas kontaktuppgifter
usp_AddKontaktuppgift	Lägger till kontaktuppgifter
usp_UpdateMedlemKontaktByID	Uppdaterar vald Kontaktuppgift
usp_GetBefattningByID	Hämtar specifik befattning
usp_AllaBefattningstyp	Listar alla befattnings typer och arvode
usp_AddBefattning	Lägger till befattning
usp_DeleteBefattning	Tar bort befattning
usp_AddMedlemTillAktivtet	Lägger till medlem till en aktivitet
usp_DeleteMedlemFronAktivitet	Tar bort medlem från aktivitet
usp_MedlemDeltarAktiviteter	Visar aktiviteter som en specifik medlem ska delta i
usp_AddAktivitet	Lägger till aktivitet, startdatum och slutdatum på aktiviteten
usp_DeleteAktivitet	Tar bort aktivitet
usp_AktivitetLista	Visar lista på aktiviteter
usp_GetSpecifikAktivitet	Visar specifik aktivitet och de medlemmar som ska delat i den

Medlem

Testar följande Lagrade Procedurer

usp_MedlemsLista

usp_GetMedlem

usp_AddMedlem

usp_DeleteMedlem

usp_UpdateMedlem

Lagrade Procedurer: usp_MedlemsLista

```
CREATE PROCEDURE usp_MedlemsLista
AS
BEGIN
            SELECT M.Fornamn, M.Efternamn, B.Befattningstyp,M.Blevmedlem
            FROM Medlem as M LEFT JOIN Befattning as B ON
M.BefattningID=B.BefattningID
            ORDER BY Fornamn ASC
END
GO
```

Funktion

Ska visa en lista med alla användare sorterar på efternamnet

Kör den lagrade proceduren

EXEC usp_MedlemsLista

Test Resultat

Visar lista på alla medlemmar och sorterar på efternamnet, lagrad procedur funkar.

Lagrade Procedurer: usp_GetMedlem

```
create PROCEDURE  appSchema.usp_getMedlem
@MedlemID int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN
                                    SELECT M.MedlemID, M.Fornamn, M.Efternamn,
M.Personnummer, M.Ort, M.Gatuadress, B.BefattningID, B.Befattningstyp, B.Arvode,
M.Blevmedlem,K.Kontaktuppgift, typ.Kontakttyp
                                    FROM Medlem as M LEFT JOIN Befattning as B ON
M.BefattningID=B.BefattningID LEFT JOIN Kontakt as K ON M.MedlemID=K.MedlemID LEFT
JOIN Kontakttyp as typ
                                    ON K.KontakttypID=typ.KontaktTypID
                                    WHERE M.MedlemID = @MedlemID
                        END
            ELSE
                        BEGIN
                        RAISERROR('Den valda medlemmen finns inte!', 16, 1)
                        END
END
```

Funktion
Ska visa detaljerad information om enskild medlem

Kör den lagrade proceduren
EXEC appSchema.usp_getMedlem 1

Test Resultat
Visar detaljerad information om enskild medlem, lagrad procedur funkar.

Lagrade Procedurer: usp_AddMedlem

```
create PROCEDURE appSchema.usp_AddMedlem

            @MedlemID int Output,
            @Personnummer varchar(12),
            @Fornamn  varchar(10),
            @Efternamn varchar(10),
            @Ort     varchar(25),
            @Gatuadress  varchar(30),
            @BefattningID int
AS
BEGIN
            BEGIN TRY
                        DECLARE @Error varchar(40)

                        BEGIN TRAN

                                    SET @Error = 'kunde inte lägga till Medlem'
                                    INSERT INTO Medlem (Personnummer,Fornamn,
Efternamn, Ort, Gatuadress, BefattningID)
                                    VALUES (@Personnummer,@Fornamn,@Efternamn
,@Ort,@Gatuadress,@BefattningID);
                                    set @MedlemID = SCOPE_IDENTITY()
                        COMMIT TRAN
                        END TRY
            BEGIN CATCH
                        ROLLBACK TRAN
                        RAISERROR(@Error, 16, 1)
            END CATCH
RETURN @MedlemID

END
```

Funktion
Ska lägga till en ny medlem i tabellen som heter medlem

Kör den lagrade proceduren
EXEC appSchema.usp_AddMedlem 0,'721213-8595', 'Emma','Peterson',
'Huddinge','Terapivägen 18 B', 3

Test Resultat
Kontrollerar tabellen medlem, ny medlem finns lagrad procedur funkar.

Lagrade Procedurer: usp_DeleteMedlem

```
CREATE PROCEDURE    appSchema.usp_DeleteMedlem
@MedlemID int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN
                                    DELETE Medlem
                                    WHERE Medlem.MedlemID = @MedlemID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda medlemmen finns inte!', 16,
1)
                        END
END
GO
```

Funktion
Ska ta bort medlem från tabellen Medlem i databasen

Kör den lagrade proceduren
EXEC appSchema.usp_DeleteMedlem 4

Test Resultat
Kontrollerar tabellen medlem, medlem med ID 4 finns inte, lagrad procedur funkar.

Lagrade Procedurer: usp_UpdateMedlem

```
create PROCEDURE appSchema.usp_UpdateMedlem
@MedlemID int = 0,
@Personnummer varchar(12),
            @Fornamn  varchar(10),
            @Efternamn varchar(10),
            @Ort     varchar(25),
            @Gatuadress  varchar(30),
            @BefattningID int
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN
                                    BEGIN TRY

                                    BEGIN TRAN

                                                UPDATE Medlem
                                                SET
            Personnummer = @Personnummer,
            Fornamn=@Fornamn,
            Efternamn=@Efternamn,
            Ort=@Ort,
            Gatuadress=@Gatuadress,
            BefattningID=@BefattningID
                                                WHERE MedlemID = @MedlemID
COMMIT TRAN
                                    END TRY
                                    BEGIN CATCH
                                    ROLLBACK TRAN
                                                RAISERROR('Ett oväntat fel
inträffade!', 16, 1)
                                    END CATCH
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda medlemmen finns inte!', 16,
1)
                        END
END
```

Funktion
Uppdaterar information på vald medlem

Kör den lagrade proceduren
EXEC appSchema.usp_UpdateMedlem 4,'721213-8595', 'Peter','Peterson',
'Huddinge','Terapivägen 17 M', 3

Test Resultat
Kontrollerar tabellen medlem, medlem med ID 4 är uppdatera, lagrad procedur funkar.

Kontakt

Testar följande Lagrade Procedurer

usp_GetMedlemKontaktByID

usp_AllaKontakttyper

usp_AddKontaktuppgift

usp_UpdateMedlemKontaktByID

Lagrade Procedurer: usp_GetMedlemKontaktByID

```
create PROCEDURE  appSchema.usp_GetMedlemKontaktByID
@MedlemID int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN


                                                                  SELECT  M.MedlemID,
k.KontaktID, K.Kontaktuppgift, typ.KontakttypID, typ.Kontakttyp
                                    FROM Medlem as M RIGHT JOIN Kontakt as K ON
M.MedlemID=K.MedlemID LEFT JOIN Kontakttyp as typ
                                    ON K.KontakttypID=typ.KontaktTypID
                              WHERE M.MedlemID = @MedlemID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda Medlemens Kontaktuppgifter
finns inte', 16, 1)
                        END
END
GO
```

Funktion

Visar kontaktinformation som en specifik medlem har

Kör den lagrade proceduren

EXEC appSchema.usp_GetMedlemKontaktByID 1

Test Resultat

Presenterar medlem med ID 1 kontakt information, lagrad procedur funkar.

Lagrade Procedurer: usp_AllaKontakttyper

```
CREATE PROCEDURE appSchema.usp_AllaKontakttyper
AS
BEGIN
            SELECT typ.KontakttypID, typ.Kontakttyp
                                    FROM Kontakttyp as typ
END
GO

EXEC appSchema.usp_AllaKontakttyper
```

Funktion
Visar alla kontakt typer som finns i tabellen kontakt typ

Kör den lagrade proceduren
EXEC appSchema.usp_AllaKontakttyper

Test Resultat
Visade all kontakttyper som finns, lagrad procedur funkar.

Lagrade Procedurer: usp_AddKontaktuppgift

```
CREATE PROCEDURE  appSchema.usp_AddKontaktuppgift
@MedlemID int = 0,
@Kontaktuppgift varchar(20),
@KontakttypID int
AS
BEGIN
            BEGIN TRY
                        INSERT INTO Kontakt (MedlemID, Kontaktuppgift,
KontakttypID)
                        VALUES (@MedlemID, @Kontaktuppgift, @KontakttypID)
            END TRY

            BEGIN CATCH
                  RAISERROR('Fel inträffade med telefonnummer/telefontyp', 16, 1)
            END CATCH
END



EXEC appSchema.usp_AddKontaktuppgift88 1,'Marco30302hotmail.com',3
```

Funktion
Lägger till ny kontakt information till specifik medlem

Kör den lagrade proceduren
EXEC appSchema.usp_AddKontaktuppgift 1,'Marco30302hotmail.com',3

Test Resultat
Kontrollerar tabellen kontakt, ny kontakt informations har lagt till medlem med ID 1, lagrad procedur funkar.

Lagrade Procedurer: usp_UpdateMedlemKontaktByID

```
Create PROCEDURE  appSchema.usp_UpdateMedlemKontaktByID
@MedlemID int = 0,
@KontaktID int = 0,
@Kontaktuppgift varchar(20)
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN

                        UPDATE Kontakt
                        set
                        Kontaktuppgift=@Kontaktuppgift
                          WHERE MedlemID = @MedlemID AND KontaktID = @KontaktID

                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda Medlemens Kontaktuppgifter
finns inte', 16, 1)
                        END
END
GO
```

Funktion
Uppdaterar en specifik kontakt information

Kör den lagrade proceduren
EXEC appSchema.usp_UpdateMedlemKontaktByID 1, 2,'olimpia'

Test Resultat
Kontrollerar tabellen kontak, kontakt information med kontakt id 2 är uppdatera, lagrad procedur funkar.

Befattning

Testar följande Lagrade Procedurer

usp_GetBefattningByID

usp_AllaBefattningstyp

usp_AddBefattning

usp_DeleteBefattning

Lagrade Procedurer: usp_GetBefattningByID

```
create PROCEDURE  appSchema.usp_GetBefattningByID
@MedlemID int = 0,
@BefattningID int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN


                                                SELECT M.MedlemID, B.BefattningID,
B.Befattningstyp, B.Arvode
                                    FROM Medlem as M RIGHT JOIN  Befattning as B ON
M.BefattningID=B.BefattningID
                                    WHERE M.MedlemID = @MedlemID AND B.BefattningID
= @BefattningID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda Medlemens Kontaktuppgifter
finns inte', 16, 1)
                        END
END
GO
```

Funktion

hämtar Befattning och arvode

Kör den lagrade proceduren

EXEC appSchema.usp_GetBefattningByID 1, 1

Test Resultat

Presenterar specifik Befattning och arvode, lagrad procedur funkar.

Lagrade Procedurer: usp_AllaBefattningstyp

```
CREATE PROCEDURE appSchema.usp_AllaBefattningstyp
AS
BEGIN

SELECT typ.BefattningID, typ.Befattningstyp
                                        FROM Befattning as typ
END
GO
```

Funktion
Visar alla befattningstyper och befattnings ID

Kör den lagrade proceduren
EXEC appSchema.usp_AllaBefattningstyp

Test Resultat
Presenterar alla befattningstyper och befattnings ID, lagrad procedur funkar.

Lagrade Procedurer: usp_AddBefattning

```
CREATE PROCEDURE usp_AddBefattning

            @Befattningstyp varchar(20),
            @Arvode int


AS

BEGIN

            BEGIN TRY
                        INSERT INTO Befattning (Befattningstyp, Arvode)
                        VALUES (@Befattningstyp, @Arvode)
            END TRY

            BEGIN CATCH
                  RAISERROR('Fel inträffade!, Befattning har inte lagts till', 16,
1)
            END CATCH

END
```

Funktion
Lägger till ny befattning och arvode
Kör den lagrade proceduren
EXEC usp_AddBefattning 'Kassör', 4000

Test Resultat
Kontrollerar tabellen befattning, ny befattning och arvode lagt till, lagrad procedur funkar.

Lagrade Procedurer: usp_DeleteBefattning

```
CREATE PROCEDURE usp_DeleteBefattning
@BefattningID int = 0
AS
BEGIN
            IF EXISTS(SELECT BefattningID FROM Befattning WHERE BefattningID =
@BefattningID)
                        BEGIN
                                    DELETE Befattning
                                    WHERE Befattning.BefattningID = @BefattningID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda Befattning finns inte!',
16, 1)
                        END
END
GO
```

Funktion
Tar bort befattning

Kör den lagrade proceduren
EXEC usp_DeleteBefattning 6

Test Resultat
Kontrollerar tabellen befattning, befattning med ID 6 är borta, lagrad procedur funkar.

Aktivtet

Testar följande Lagrade Procedurer

usp_AddMedlemTillAktivtet

usp_DeleteMedlemFronAktivitet

usp_MedlemDeltarAktiviteter

usp_AddAktivitet

usp_DeleteAktivitet

usp_AktivitetLista

usp_GetSpecifikAktivitet

Lagrade Procedurer: usp_AddMedlemTillAktivtet

```
CREATE PROCEDURE usp_AddMedlemTillAktivtet
@MedlemID int = 0,
@AktivitetstypID int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        AND EXISTS(SELECT AktivitetstypID FROM Aktivitetstyp WHERE
AktivitetstypID = @AktivitetstypID)
                        BEGIN
                                    BEGIN TRY
                                                INSERT INTO Aktivitet (MedlemID,
AktivitetstypID)
                                                VALUES (@MedlemID,
@AktivitetstypID)
                                    END TRY
                                    BEGIN CATCH
                                                RAISERROR('Ett oväntat fel
inträffade! Försök igen.', 16, 1)
                                    END CATCH
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda medlemmen/aktiviteten
finns inte! Försök igen.', 16, 1)
                        END
END
GO
```

Funktion

Läger till medlem i aktivitet

Kör den lagrade proceduren

EXEC usp_AddMedlemTillAktivtet 5,2

Test Resultat
Kontrollerar tabellen aktivitet, aktivitet med ID 5 har lagts till aktivitet med id 2, lagrad procedur funkar.

Lagrade Procedurer: usp_DeleteMedlemFronAktivitet

```
CREATE PROCEDURE   usp_DeleteMedlemFronAktivitet
@MedlemID int = 0,
@Aktivitettyp int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Aktivitet WHERE MedlemID = @MedlemID)
                        BEGIN
                                    DELETE Aktivitet
                                    WHERE Aktivitet.MedlemID = @MedlemID AND
Aktivitet.Aktivitettyp = @Aktivitettyp
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda medlemmen finns inte!',
16, 1)
                        END
END
GO
```

Funktion
Tar bort medlem från aktivitet

Kör den lagrade proceduren
EXEC usp_DeleteAktivitet 5,2

Test Resultat
Kontrollerar tabellen aktivitet, medlem med ID 5 har tagit bort från aktivitet med id 2, lagrad procedur funkar.

Lagrade Procedurer: usp_MedlemDeltarAktiviteter

```
CREATE PROCEDURE  usp_MedlemDeltarAktiviteter
@medlemID int = 0
AS
BEGIN
            IF EXISTS(SELECT MedlemID FROM Medlem WHERE MedlemID = @MedlemID)
                        BEGIN

SELECT M.Fornamn, M.Efternamn, typ.Aktivitetstyp
                                    FROM Medlem as M LEFT JOIN Aktivitet as A ON
M.medlemID=A.medlemID LEFT JOIN Aktivitetstyp as typ ON
A.AktivitetstypID=typ.AktivitetstypID
                                    WHERE M.medlemID = @medlemID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda medlemmen finns inte!',
16, 1)
                        END
END
GO
```

Funktion
Visar aktivteter en specifik medlem ska delta i

Kör den lagrade proceduren
EXEC usp_MedlemDeltarAktiviteter 2

Test Resultat
Lagrad procedur presenterar aktivteter en specifik medlem ska delta i, lagrad procedur funkar.

Lagrade Procedurer: usp_AddAktivitet

```
CREATE PROCEDURE usp_AddAktivitet

@Aktivitetstyp  varchar(15),
@Startdatum date,
@Slutdatum date

AS

BEGIN

            BEGIN TRY
                        INSERT INTO Aktivitetstyp (Aktivitetstyp, Startdatum,
Slutdatum)
                        VALUES (@Aktivitetstyp, @Startdatum, @Slutdatum)
            END TRY

            BEGIN CATCH
                  RAISERROR('Fel inträffade!,Aktivitet har inte lagts till', 16, 1)
            END CATCH

END
```

Funktion
Lägger till aktivitet i tabellen aktivitetstyp

Kör den lagrade proceduren
EXEC usp_AddAktivitet 'Musik', '2015-04-25', '2015-06-25'

Test Resultat
Kontrollerar tabellen aktivitetstyp, ny aktivitet har lagt till, lagrad procedur funkar.

Lagrade Procedurer: usp_DeleteAktivitet

```
CREATE PROCEDURE   usp_DeleteAktivitet
@AktivitetstypID int = 0
AS
BEGIN
            IF EXISTS(SELECT AktivitetstypID FROM Aktivitetstyp WHERE
AktivitetstypID = @AktivitetstypID)
                        BEGIN
                                    DELETE Aktivitetstyp
                                    WHERE Aktivitetstyp.AktivitetstypID =
@AktivitetstypID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda aktivteten finns inte!',
16, 1)
                        END
END
GO
```

Funktion
Tar bort aktivitet i tabellen aktivitetstyp

Kör den lagrade proceduren
EXEC usp_DeleteAktivitet 5

Test Resultat
Kontrollerar tabellen aktivitetstyp, aktivitet har tagit bort, lagrad procedur funkar.

Lagrade Procedurer: usp_AktivitetLista

```
CREATE PROCEDURE usp_AktivitetLista
AS
BEGIN
            SELECT Aktivitetstyp, Startdatum, Slutdatum
            FROM Aktivitetstyp
            ORDER BY Aktivitetstyp ASC
END
GO
```

Funktion
Visar lista på alla aktivteter

Kör den lagrade proceduren
EXEC usp_AktivitetLista

Test Resultat
Lagrad procedur presenterar alla aktivteter, lagrad procedur funkar.

Lagrade Procedurer: usp_GetSpecifikAktivitet

```
CREATE PROCEDURE  usp_GetSpecifikAktivitet
@AktivitetstypID int = 0
AS
BEGIN
            IF EXISTS(SELECT AktivitetstypID FROM Aktivitetstyp WHERE
AktivitetstypID = @AktivitetstypID)
                        BEGIN
                                    SELECT  M.medlemID, M.Fornamn, M.Efternamn,
M.Personnummer, typ.Aktivitetstyp
                                    FROM Medlem as M LEFT JOIN Aktivitet as A ON
M.medlemID=A.medlemID LEFT JOIN Aktivitetstyp as typ ON
A.AktivitetstypID=typ.AktivitetstypID
                                    WHERE typ.AktivitetstypID = @AktivitetstypID
                        END
            ELSE
                        BEGIN
                                    RAISERROR('Den valda medlemmen finns inte!',
16, 1)
                        END
END
GO

EXEC usp_GetSpecifikAktivitet 2
```

Funktion

Visar aktivitet och information om medlemmar som deltar

Kör den lagrade proceduren

```
EXEC usp_GetSpecifikAktivitet 2
```

Test Resultat

Lagrad procedur presenterar en specifik aktivitet och de medlemmar som ska delta på den, lagrad procedur funkar.