**Test av Lagrade Procedurer**

**Testmål**

|  |  |
| --- | --- |
| **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 |

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

**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, typ.arvode  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

usp\_UpdateMedlemKontaktByID

**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 appSchema.usp\_DeleteMedlemFronAktivitet  @MedlemID int = 0,  @AktivitetstypID int = 0  AS  BEGIN  IF EXISTS(SELECT MedlemID FROM Aktivitet WHERE MedlemID = @MedlemID)  BEGIN  DELETE Aktivitet  WHERE Aktivitet.MedlemID = @MedlemID AND Aktivitet.AktivitetstypID = @AktivitetstypID  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.

**Lagrade Procedurer: usp\_UpdateMedlemKontaktByID**

|  |
| --- |
| alter PROCEDURE appSchema.usp\_UpdateMedlemKontaktByID  @AktivitetstypID int = 0,  @Aktivitetstyp varchar(15),  @Startdatum date,  @Slutdatum date  AS  BEGIN  IF EXISTS(SELECT AktivitetstypID FROM Aktivitetstyp WHERE AktivitetstypID = @AktivitetstypID)  BEGIN    UPDATE Aktivitetstyp  set  Aktivitetstyp=@Aktivitetstyp,  Startdatum=@Startdatum,  Slutdatum=@Slutdatum  WHERE AktivitetstypID = @AktivitetstypID  END  ELSE  BEGIN  RAISERROR('Den valda Medlemens Kontaktuppgifter finns inte', 16, 1)  END  END  GO  EXEC appSchema.usp\_UpdateMedlemKontaktByID |

**Funktion**

Uppdaterar aktivitet

**Kör den lagrade proceduren**

EXEC appSchema.usp\_UpdateMedlemKontaktByID 2, 'musik', 2015-06-07, 2015-07-06

**Test Resultat**

Lagrad procedur uppdaterar aktivitet, lagrad procedur funkar.

**Lagrade Procedurer som kontrollerar**

**Lagrade Procedurer: appSchema.aktivitetMedlemexisterar**

|  |
| --- |
| create PROCEDURE appSchema.aktivitetMedlemexisterar    @medlemID int = 0,  @AktivitetstypID int = 0,  @intOutput int OUTPUT  AS  if exists(SELECT \* FROM Aktivitet Where medlemID = @medlemID and AktivitetstypID= @AktivitetstypID)  begin  set @intOutput = 1;  end  else  begin  set @intOutput = 0;  end  RETURN @intOutput  go |

**Funktion**

Kontrollerar om medlem finns i aktivitet, om den existerar i aktivitet får man 1 om den inte finns så får man 0

**Kör den lagrade proceduren**

EXEC appSchema.aktivitetMedlemexisterar 2, 5

**Test Resultat**

Lagrad procedur medlem finns i aktivitet , lagrad procedur funkar.