M153 Projekt

Trainingsplan Datenbank

Inhaltsverzeichnis

Einleitung

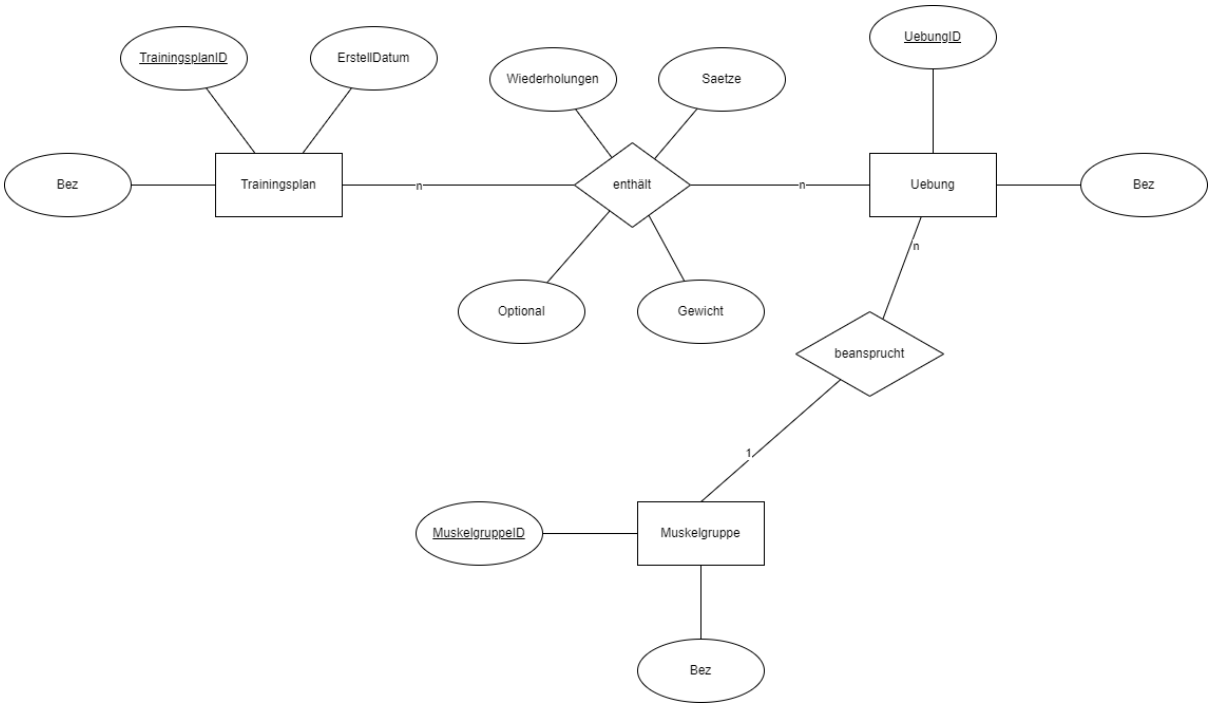
Kurzbeschreibung

Die Datenbank dient zur Speicherung von Trainingsplänen. Ein Trainingsplan enthält verschiedene Übungen, welche dem Sportler zu einem gesunden Körper und Lebensweise verhelfen. Die Anzahl Sätze und Wiederholungen der jeweiligen Übungen ist je nach Trainingsplan unterschiedlich. Jede Übung hat eine Muskelgruppe zugewiesen, damit es besser filtrierbar ist und damit der Sportler den Nutzen der Übung versteht. Die Trainingspläne sollen einfach exportiert und auch verwendet werden können.

Rapport

Der Rapport wurde mit GitHub realisiert.

ER Diagramm



Relationales Modell

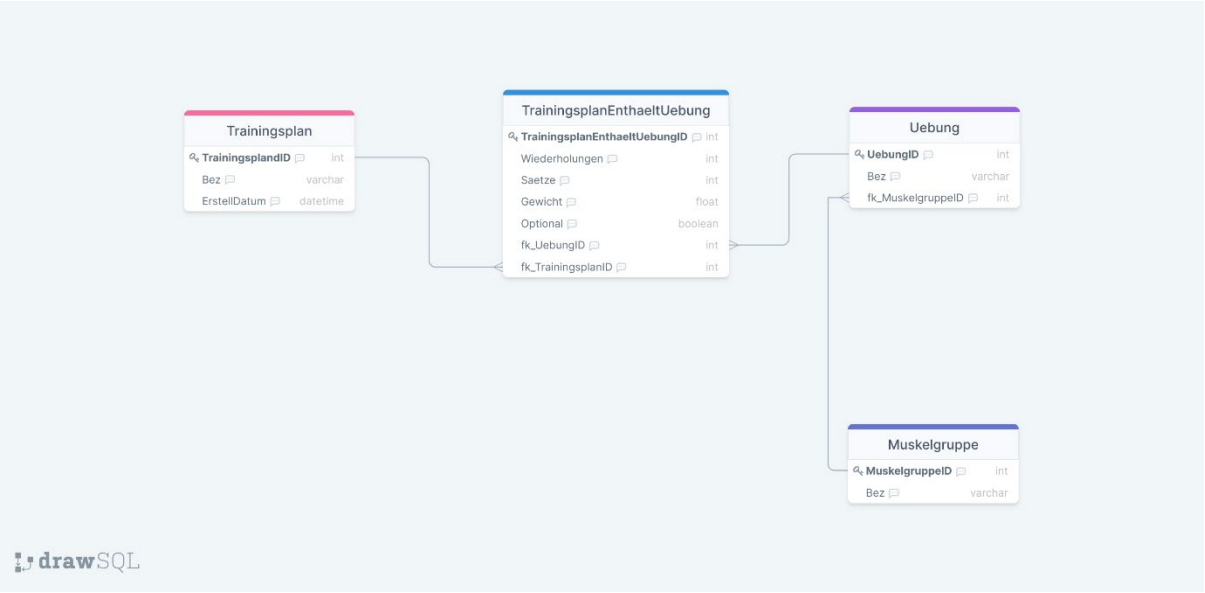
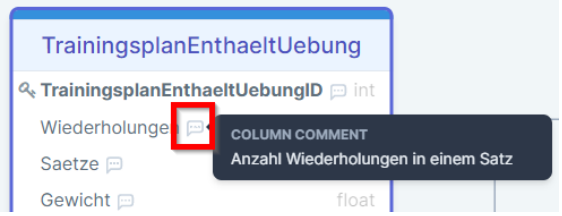


Diagramm mit Beschreibung der Felder:
https://drawsql.app/trainingsplandb/diagrams/trainingsplan/embed



Abfragen/Manipulationen

Alle Übungen des Ganzkörperplans anzeigen

```
SELECT
Uebung.Bez as 'Übung',
TrainingsplanEnthaeltUebung.Saetze AS 'Sätze',
TrainingsplanEnthaeltUebung.Wiederholungen AS 'Wiederholungen',
TrainingsplanEnthaeltUebung.Gewicht AS 'Gewicht',
TrainingsplanEnthaeltUebung.OptiONal AS 'Optional',
Muskelgruppe.Bez AS 'Muskelgruppe'
FROM Trainingsplan
JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_TrainingsplanID = Traini
ngsplan.TrainingsplandID
JOIN Uebung ON Uebung.UebungID = TrainingsplanEnthaeltUebung.fk_UebungID
JOIN Muskelgruppe ON Muskelgruppe.MuskelgruppeID = Uebung.fk_MuskelgruppeID
WHERE Trainingsplan.Bez = 'Ganzkörper'
GO
```

Mit dieser Abfrage werden alle Übungen mit allen relevanten Informationen eines Trainingplans angezeigt.

Übungen ohne Plan

```
SELECT UebungID, Uebung.Bez as 'Übung' FROM Uebung
LEFT JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebung.U
ebungID
WHERE TrainingsplanEnthaeltUebung.TrainingsplanEnthaeltUebungID IS NULL
GO
```

Mit dieser Abfrage werden alle Übungen angezeigt, welche keinem Trainingsplan zugewiesen sind.

Übung updaten

```
SELECT * FROM Uebung WHERE Bez = 'Kniebeugen'
GO
UPDATE Uebung SET fk_MuskelgruppeID = 2 WHERE Bez = 'Kniebeugen'
GO
SELECT * FROM Uebung WHERE Bez = 'Kniebeugen'
GO
```

Mit dieser Manipulation wird die Muskelgruppe einer Übung geändert.

Übung updaten schlägt fehl wegen Trigger

```
UPDATE Uebung SET Bez = 'Ausfallschritt' WHERE Bez = 'Kniebeugen'
GO
```

Da es bereits eine Übung mit der Bezeichnung „Ausfallschritt“ existiert, wirft unser Trigger einen Error.

```
Msg 50000, Level 11, State 10, Procedure CheckUebungOnUpdate, Line 13
Folgende Übungen sind bereits vorhanden : Ausfallschritt
```

Übersicht aller Trainingspläne

```
SELECT
Trainingsplan.Bez as [Trainingsplan],
Trainingsplan.ErstellDatum as [Erstellt am],
dbo.GetAllMuskelGruppenOfTrainingsplan(Trainingsplan.TrainingsplandID) AS [Muskelgruppen]
FROM Trainingsplan
GO
```

Mit dieser Abfrage werden alle Trainingspläne angezeigt. Mit unserer selbstentwickelten Funktion wird zusätzlich noch angezeigt, welche Muskelgruppen wie oft vorkommen.

Trainingsplan erstellen mit stored Procedure

```
EXEC sp_CreateTrainingplan 'TestPlan';
SELECT * FROM Trainingsplan Where Bez = 'TestPlan'
```

Das Erstellen des Trainingsplan mithilfe der eigenentwickelten stored Procedure „sp_CreateTrainingsplan“.

Gespeicherte Prozeduren

Sp_CreateTrainingsplan

```
DROP PROCEDURE IF exists sp_CreateTrainingplan;
go

CREATE PROCEDURE sp_CreateTrainingplan
    @bez varchar(50)
AS
BEGIN
    INSERT INTO Trainingsplan
        (Bez, ErstellDatum)
    VALUES(@bez, GETDATE());
    RETURN 1;
END
GO
```

Mithilfe der Prozedur „sp_CreateTrainingsplan“ können Trainingspläne einfacher erstellt werden. Beim Ausführen der Prozedur muss jede glich die gewünschte Bezeichnung für den Trainingsplan als Parameter übergeben werden. Das Erstelldatum wird automatisch mit dem momentanen Systemdatum abgefüllt.

Gespeicherte Funktionen

GetAllMuskelGruppenOfTrainingsplan

```
Drop FUNCTION IF EXISTS dbo.GetAllMuskelGruppenOfTrainingsplan;
GO
CREATE FUNCTION dbo.GetAllMuskelGruppenOfTrainingsplan (@trainingsPlanId INT)
RETURNS VARCHAR(MAX)
AS
BEGIN
    DECLARE @Result VARCHAR(max)
    DECLARE @Enumerator TABLE (id INT)

    INSERT INTO @Enumerator
    SELECT DISTINCT Muskelgruppe.MuskelgruppeID
    FROM Muskelgruppe
        JOIN Uebung ON Uebung.fk_MuskelgruppeID = Muskelgruppe.MuskelgruppeID
        JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebun
g.UebungID
    WHERE fk_TrainingsplanID = @trainingsPlanId

    DECLARE @id INT
    WHILE EXISTS (SELECT 1
    FROM @Enumerator)
    BEGIN
        SELECT TOP 1
            @id = id
        FROM @Enumerator

        SET @Result = CONCAT(
        @Result,
        ', ',
        (SELECT Muskelgruppe.Bez FROM Muskelgruppe WHERE Muskelgruppe.MuskelgruppeID = @id)
,
        '(',
        (SELECT COUNT(Uebung.fk_MuskelgruppeID) FROM Uebung JOIN TrainingsplanEnthaeltUebun
g ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebung.UebungID
        WHERE TrainingsplanEnthaeltUebung.fk_TrainingsplanID = @trainingsPlanId
        AND Uebung.fk_MuskelgruppeID = @id),
        ')')

        DELETE FROM @Enumerator WHERE id = @id
    END
```

Die Funktion „GetAllMuskelGruppenOfTrainingsplan“ gibt einen String zurück, mit allen Muskeln Gruppen, die in einem Trainingsplan vorkommen. Zu jeder Muskelgruppe wird noch angegeben, wie oft diese im Trainingsplan vorkommt. Als Parameter erhält die Funktion die ID des Trainingsplan. Anhand dieser ID werden alle IDs der verwendeten Muskelgruppen in die Variabel @Enumerator gespeichert.

```
INSERT INTO @Enumerator
    SELECT DISTINCT Muskelgruppe.MuskelgruppeID
    FROM Muskelgruppe
        JOIN Uebung ON Uebung.fk_MuskelgruppeID = Muskelgruppe.MuskelgruppeID
        JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebun
g.UebungID
    WHERE fk_TrainingsplanID = @trainingsPlanId
```

Durch diese IDs wird iteriert und dem Result-String die Bezeichnung, sowie die Anzahl der Muskelgruppe hinzugefügt („[Bezeichnung]([Anzahl]), “). Der Result-String wird anschliessend zurückgegeben und die letzten zwei Zeichen entfernt („, “).

Trigger

CheckUebungOnInsert

```
DROP trigger IF exists CheckUebungOnInsert;
go

CREATE TRIGGER CheckUebungOnInsert ON Uebung INSTEAD OF INSERT
AS BEGIN
    DECLARE @AnzahlDoppelteUebungen INT = (SELECT COUNT(*) as Anz FROM (SELECT Uebung.Bez F
ROM Uebung INTERSECT SELECT inserted.bez FROM inserted) I);

    IF (@AnzahlDoppelteUebungen > 0)
    begin
        DECLARE @DoppelteUebungen VARCHAR(max) = (SELECT STRING_AGG (Bez, ',') AS bezeichnu
ngen FROM (SELECT Uebung.Bez FROM Uebung INTERSECT SELECT inserted.bez FROM inserted) I)
        RAISERROR('Folgende Übungen sind bereits vorhanden : %s', 11, 10, @DoppelteUebungen
);
    END
    else
        INSERT INTO Uebung (Bez, fk_MuskelgruppeID)
    SELECT inserted.Bez, inserted.fk_MuskelgruppeID
    FROM inserted
END
GO
```

Der Trigger „CheckUebungenOnInsert“ prüft beim Erstellen einer neuen Übung, ob bereits eine Übung mit derselben Bezeichnung existiert. Damit INSERT mehrere Datensätze gleichzeitig eingefügt werden können, wird mithilfe von INTERSECT geprüft, ob Übungen in der INSERTED, sowie in der „Uebung“ Tabelle vorkommen. Da die INSERTED Tabelle mit der Uebung Tabelle ohne die neuen Werte verglichen werden soll, arbeiten wir beim Trigger nicht mit FOR sondern mit INSEAD OF. Daher müssen wir die Daten am Schluss des Triggers noch manuell einfügen. Falls doppelte Bezeichnungen vorkommen, werfen wir einen Error mit einer Liste aller doppelten Übungen. Die Liste erstellen wir mit der „STRING_AGG“ Funktion.

CheckUebungOnUpdate

```
DROP trigger IF exists CheckUebungOnUpdate;
go

CREATE TRIGGER CheckUebungOnUpdate ON Uebung INSTEAD OF UPDATE
AS BEGIN
    DECLARE @AnzahlDoppelteUebungen INT = (SELECT COUNT(*) as Anz FROM (SELECT Uebung.Bez F
ROM Uebung INTERSECT SELECT inserted.bez FROM inserted) I);

    DECLARE @Bez VARCHAR(MAX), @fk_MuskelgruppeId INT, @UebungId INT
    SELECT @Bez = INSERTED.bez, @fk_MuskelgruppeId = INSERTED.fk_MuskelgruppeID, @UebungId
= INSERTED.UebungId
    FROM INSERTED

    IF (@AnzahlDoppelteUebungen > 0 AND UPDATE(Bez))
    begin
        DECLARE @DoppelteUebungen VARCHAR(max) = (SELECT STRING_AGG (Bez, ',') AS bezeichnu
ngen FROM (SELECT Uebung.Bez FROM Uebung INTERSECT SELECT inserted.bez From inserted) I)
        RAISERROR('Folgende Übungen sind bereits vorhanden : %s', 11, 10, @DoppelteUebungen
);
    END
    else
        UPDATE Uebung SET Uebung.bez = @bez, Uebung.fk_MuskelgruppeId = @fk_MuskelgruppeId
WHERE UebungId = @UebungId;
END
GO
```

Der Trigger „CheckUebungOnUpdate“ hat dieselbe funktionsweise wie CheckUebungOnInsert, allerdings beim Updaten einer Übung.

Anhang

M153_Projekt_Create.sql

```
USE Master;
GO
DROP DATABASE IF EXISTS Trainingsplan
GO
CREATE DATABASE Trainingsplan
GO
USE Trainingsplan
GO


CREATE TABLE Trainingsplan
(
    TrainingsplandID INT NOT NULL IDENTITY PRIMARY KEY,
    Bez VARCHAR(255) NOT NULL ,
    ErstellDatum DATETIME NOT NULL
);
CREATE TABLE Uebung
(
    UebungID INT NOT NULL IDENTITY PRIMARY KEY,
    Bez VARCHAR(255) NOT NULL ,
    fk_MuskelgruppeID INT NOT NULL
);
CREATE TABLE Muskelgruppe
(
    MuskelgruppeID INT NOT NULL IDENTITY PRIMARY KEY,
    Bez VARCHAR(255) NOT NULL
);
CREATE TABLE TrainingsplanEnthaeltUebung
(
    TrainingsplanEnthaeltUebungID INT NOT NULL IDENTITY PRIMARY KEY,
    Wiederholungen INT NOT NULL,
    Saetze INT NOT NULL,
    Gewicht DECIMAL(8, 2) NOT NULL,
    Optional BIT NOT NULL,
    fk_UebungID INT NOT NULL ,
    fk_TrainingsplanID INT NOT NULL
);
GO
ALTER TABLE
Uebung ADD CONSTRAINT uebung_fk_muskelgruppeid_foreign FOREIGN KEY(fk_MuskelgruppeID) REFER
ENCES Muskelgruppe(MuskelgruppeID);
ALTER TABLE
TrainingsplanEnthaeltUebung ADD CONSTRAINT trainingsplanenthaeltuebung_fk_uebungid_foreign
FOREIGN KEY(fk_UebungID) REFERENCES Uebung(UebungID);
ALTER TABLE
TrainingsplanEnthaeltUebung ADD CONSTRAINT trainingsplanenthaeltuebung_fk_trainingsplanid_f
oreign FOREIGN KEY(fk_TrainingsplanID) REFERENCES Trainingsplan(TrainingsplandID);
GO


/* Trigger erstellen */
DROP trigger if exists CheckUebungOnInsert;
go
```

```
CREATE TRIGGER CheckUebungOnInsert ON Uebung INSTEAD OF INSERT
AS BEGIN
    DECLARE @AnzahlDoppelteUebungen INT = (SELECT COUNT(*) as Anz
    FROM (                          SELECT Uebung.Bez
            FROM Uebung
        INTERSECT
            SELECT inserted.bez
            From inserted) I);

    if (@AnzahlDoppelteUebungen > 0)
    begin
        DECLARE @DoppelteUebungen VARCHAR(max) = (SELECT STRING_AGG (Bez, ',') AS bezeichnu
ngen
        FROM (                                  SELECT Uebung.Bez
                FROM Uebung
            INTERSECT
                SELECT inserted.bez
                From inserted) I)
        raiserror('Folgende Übungen sind bereits vorhanden : %s', 11, 10, @DoppelteUebungen
);
    END
    else
        Insert INTO Uebung
        (Bez, fk_MuskelgruppeID)
    Select inserted.Bez, inserted.fk_MuskelgruppeID
    FROM inserted
END
GO

DROP trigger if exists CheckUebungOnUpdate;
go

CREATE TRIGGER CheckUebungOnUpdate ON Uebung INSTEAD OF UPDATE
AS BEGIN
    DECLARE @AnzahlDoppelteUebungen INT = (SELECT COUNT(*) as Anz
    FROM (                          SELECT Uebung.Bez
            FROM Uebung
        INTERSECT
            SELECT inserted.bez
            From inserted) I);

    DECLARE @Bez VARCHAR(MAX), @fk_MuskelgruppeId INT, @UebungId INT
    SELECT @Bez = INSERTED.bez, @fk_MuskelgruppeId = INSERTED.fk_MuskelgruppeID, @UebungId
= INSERTED.UebungId
    FROM INSERTED


    if (@AnzahlDoppelteUebungen > 0 AND UPDATE(Bez))
    begin
        DECLARE @DoppelteUebungen VARCHAR(max) = (SELECT STRING_AGG (Bez, ',') AS bezeichnu
ngen
        FROM (                                  SELECT Uebung.Bez
                FROM Uebung
            INTERSECT
                SELECT inserted.bez
                From inserted) I)
        raiserror('Folgende Übungen sind bereits vorhanden : %s', 11, 10, @DoppelteUebungen
);
```

```
    END
    else
        UPDATE Uebung SET Uebung.bez = @bez, Uebung.fk_MuskelgruppeId = @fk_MuskelgruppeId
WHERE UebungId = @UebungId;
END
GO


/* Funktion erstellen */
Drop FUNCTION IF EXISTS dbo.GetAllMuskelGruppenOfTrainingsplan;
GO
CREATE FUNCTION dbo.GetAllMuskelGruppenOfTrainingsplan (@trainingsPlanId INT)
RETURNS VARCHAR(MAX)
AS
BEGIN
    declare @Result VARCHAR(max)
    declare @Enumerator TABLE (id INT)

    INSERT INTO @Enumerator
    SELECT DISTINCT Muskelgruppe.MuskelgruppeID
    FROM Muskelgruppe
        JOIN Uebung ON Uebung.fk_MuskelgruppeID = Muskelgruppe.MuskelgruppeID
        JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebun
g.UebungID
    WHERE fk_TrainingsplanID = @trainingsPlanId

    DECLARE @id INT
    WHILE EXISTS (SELECT 1
    FROM @Enumerator)
    BEGIN
        SELECT TOP 1
            @id = id
        FROM @Enumerator

        SET @Result = CONCAT(
        @Result,
        ', ',
        (SELECT Muskelgruppe.Bez FROM Muskelgruppe WHERE Muskelgruppe.MuskelgruppeID = @id)
,
        '(',
        (SELECT COUNT(Uebung.fk_MuskelgruppeID) FROM Uebung JOIN TrainingsplanEnthaeltUebun
g ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebung.UebungID
        WHERE TrainingsplanEnthaeltUebung.fk_TrainingsplanID = @trainingsPlanId
        AND Uebung.fk_MuskelgruppeID = @id),
        ')')

        DELETE FROM @Enumerator WHERE id = @id
    END

    RETURN right(@Result, len(@Result)-2);
END;
GO


/* Prozedur Trainingsplan erstellen */
DROP PROCEDURE if exists sp_CreateTrainingplan;
go
```

```
CREATE PROCEDURE sp_CreateTrainingplan
    @bez varchar(50)
AS
BEGIN
    INSERT INTO Trainingsplan
        (Bez, ErstellDatum)
    VALUES(@bez, GETDATE());
    RETURN 1;
END
GO
```

M153_Projekt_Insert.sql

```
Use Trainingsplan
Go

DELETE FROM Muskelgruppe;
DELETE FROM Trainingsplan;
DELETE FROM TrainingsplanEnthaeltUebung;
DELETE FROM Uebung;
GO

INSERT INTO Muskelgruppe (Bez)
VALUES
('Beine'),
('Rücken'),
('Bauch'),
('Arme'),
('Schultern'),
('Brust')
GO

DECLARE @BeineID INT = (SELECT MuskelgruppeId From Muskelgruppe WHERE Muskelgruppe.Bez = 'B
eine');
DECLARE @RückenID INT = (SELECT MuskelgruppeId From Muskelgruppe WHERE Muskelgruppe.Bez = '
Rücken');
DECLARE @BauchID INT = (SELECT MuskelgruppeId From Muskelgruppe WHERE Muskelgruppe.Bez = 'B
auch');
DECLARE @ArmeID INT = (SELECT MuskelgruppeId From Muskelgruppe WHERE Muskelgruppe.Bez = 'Ar
me');
DECLARE @SchulternID INT = (SELECT MuskelgruppeId From Muskelgruppe WHERE Muskelgruppe.Bez
= 'Schultern');
DECLARE @BrustID INT = (SELECT MuskelgruppeId From Muskelgruppe WHERE Muskelgruppe.Bez = 'B
rust');

INSERT INTO Uebung (Bez, fk_MuskelgruppeID)
VALUES
('Kniebeugen', @BeineID),
('Ausfallschritt', @BeineID),
('Beinheben', @BeineID),
('Kreuzheben', @BeineID),
('Wadenheben', @BeineID),

('Langhantel-Rudern', @RückenID),
('Kurzhantel-Rudern', @RückenID),
('Latzug', @RückenID),
('Klimmzug', @RückenID),
('Überzüge', @RückenID),

('Crunch', @BauchID),
('Plank', @BauchID),
('Side Plank', @BauchID),
('Russian Twist', @BauchID),
('Mountain Climber', @BauchID),

('Trizepsdrücken', @ArmeID),
('Dips', @ArmeID),
('Arnold Dips', @ArmeID),
('Enges Bankdrücken', @ArmeID),
```

```
('Curls', @ArmeID),

('Butterfly Reverse' ,@SchulternID),
('Schulterdrücken', @SchulternID),
('Seitenheben', @SchulternID),
('Frontheben', @SchulternID),
('Military Press', @SchulternID),

('Bankdrücken', @BrustID),
('Liegestütz', @BrustID),
('Fliegende', @BrustID),
('Schrägbankdrücken', @BrustID),
('Kabelzug-Fliegende', @BrustID)
GO

INSERT INTO Trainingsplan (Bez, ErstellDatum)
VALUES
('Ganzkörper', '2020-01-03'),
('Leg day', '2020-04-02'),
('Oberkörper', '2021-06-09')
GO

DECLARE @GanzKörperPlan INT = (SELECT TrainingsplandID From Trainingsplan WHERE Trainingspl
an.Bez = 'Ganzkörper');
DECLARE @LegDayPlan INT = (SELECT TrainingsplandID From Trainingsplan WHERE Trainingsplan.B
ez = 'Leg day');
DECLARE @OberkörperPlan INT = (SELECT TrainingsplandID From Trainingsplan WHERE Trainingspl
an.Bez = 'Oberkörper');
INSERT INTO TrainingsplanEnthaeltUebung (fk_TrainingsplanID, fk_UebungID, Saetze, Wiederhol
ungen, Gewicht, Optional)
VALUES
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Kniebeugen'), 3, 10, 120
, 0),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Kreuzheben'), 5, 6, 150,
 0),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Klimmzug'), 3, 10, 0, 0)
,
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Überzüge'), 3, 10, 70, 0
),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Crunch'), 2, 25, 0, 0),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Dips'), 4, 12, 0, 0),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Schulterdrücken'), 3, 10
, 40, 0),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Bankdrücken'), 3, 10, 10
, 0),
(@GanzKörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Liegestütz'), 3, 50, 0,
1),

(@LegDayPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Kniebeugen'), 5, 8, 150, 0),
(@LegDayPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Ausfallschritt'), 3, 10, 60,
 0),
(@LegDayPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Beinheben'), 3, 12, 70, 0),
(@LegDayPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Kreuzheben'), 5, 8, 180, 0),
(@LegDayPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Wadenheben'), 3, 12, 40, 0),

(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Latzug'), 3, 12, 80, 0),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Langhantel-
Rudern'), 3, 10, 65, 0),
```

```
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Trizepsdrücken'), 3, 10,
 20, 0),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Curls'), 3, 10, 18, 0),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Seitenheben'), 3, 12, 40
, 0),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Schulterdrücken'), 3, 12
, 50, 0),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Bankdrücken'), 3, 12, 90
, 0),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Fliegende'), 3, 12, 20,
1),
(@OberkörperPlan, (Select UebungId From Uebung Where Uebung.Bez = 'Kabelzug-
Fliegende'), 3, 12, 20, 1)
```

M153_Projekt_Query.sql

```
/* Alle Übungen des Ganzkörperplans */
SELECT
Uebung.Bez as 'Übung',
TrainingsplanEnthaeltUebung.Saetze AS 'Sätze',
TrainingsplanEnthaeltUebung.Wiederholungen AS 'Wiederholungen',
TrainingsplanEnthaeltUebung.Gewicht AS 'Gewicht',
TrainingsplanEnthaeltUebung.OptiONal AS 'Optional',
Muskelgruppe.Bez AS 'Muskelgruppe'
FROM Trainingsplan
JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_TrainingsplanID = Traini
ngsplan.TrainingsplandID
JOIN Uebung ON Uebung.UebungID = TrainingsplanEnthaeltUebung.fk_UebungID
JOIN Muskelgruppe ON Muskelgruppe.MuskelgruppeID = Uebung.fk_MuskelgruppeID
WHERE Trainingsplan.Bez = 'Ganzkörper'
GO

/* Übungen, welche keinem Plan zugewiesen sind anzeigen */

SELECT UebungID, Uebung.Bez as 'Übung' FROM Uebung
LEFT JOIN TrainingsplanEnthaeltUebung ON TrainingsplanEnthaeltUebung.fk_UebungID = Uebung.U
ebungID
WHERE TrainingsplanEnthaeltUebung.TrainingsplanEnthaeltUebungID IS NULL
GO

/* Eine Übung updaten */
SELECT * FROM Uebung WHERE Bez = 'Kniebeugen'
GO
UPDATE Uebung SET fk_MuskelgruppeID = 2 WHERE Bez = 'Kniebeugen'
GO
SELECT * FROM Uebung WHERE Bez = 'Kniebeugen'
GO

/* Eine Übung updaten wenn die neue Bezeichnung bereits existiert */
/* -> Sollte fehlschlagen, da Ausfallschritt bereits exisitert*/
SELECT * FROM Uebung WHERE Bez = 'Kniebeugen'
GO
UPDATE Uebung SET Bez = 'Ausfallschritt' WHERE Bez = 'Kniebeugen'
GO
SELECT * FROM Uebung WHERE Bez = 'Kniebeugen'
GO

/* Übersicht aller Plane*/
SELECT Trainingsplan.Bez as [Trainingsplan], Trainingsplan.ErstellDatum as [Erstellt am], d
bo.GetAllMuskelGruppenOfTrainingsplan(Trainingsplan.TrainingsplandID) AS [Muskelgruppen] FR
OM Trainingsplan
GO

/* Ein Trainingsplan via Stored Procedure erstellen und danach ausgeben */
DECLARE @bez varchar(55) = 'Pull Day'
EXEC sp_CreateTrainingplan @bez
SELECT * FROM Trainingsplan WHERE Trainingsplan.Bez = @bez
```