**20.10.2014**

**1 Aufgabenkomplex Praktikum SQL – MS SQL - Grundlagen**

**2.1**

SELECT \* FROM Mitarbeiter;

SELECT \* FROM Zuordnung;

SELECT \* FROM Projekt;

ALTER TABLE Projekt ADD LeiterID CHAR(40);

**2.2**

UPDATE Projekt SET LeiterID = (SELECT MitID FROM Mitarbeiter m WHERE m.Nachname=Projekt.ProLeiter);

**2.3**

UPDATE Projekt SET LeiterID = (SELECT MitID FROM Mitarbeiter WHERE Vorname = 'Fred')

WHERE LeiterID IS NULL AND ProLeiter IS NOT NULL;

**2.4**

ALTER TABLE Projekt DROP COLUMN ProLeiter;

**3.1**

INSERT INTO Mitarbeiter(MitID, Nachname, Vorname, Ort, Gebdat, Beruf, Telnr) VALUES ('210', 'Merko', 'Anxhela', 'Dresden', '1996-03-02', 'Student', '01746252684');

**3.2**

INSERT INTO Projekt(ProNr, ProName, ProOrt, ProBeschreibung, ProAufwand, LeiterID) VALUES ('46', 'MS-SQL-Prakt', 'Dresden', '', '3', '210');

UPDATE Projekt SET ProBeschreibung = NULL WHERE LeiterID=210;

**3.3**

INSERT INTO Zuordnung (MitID, ProNr, Istanteil, Plananteil) VALUES ('210', '46', 0.8, 0.65);

**3.4**

INSERT INTO Zuordnung (MitID, ProNr, Istanteil, Plananteil) VALUES ('210', '31', 0.65, 0.35);

**3.5**

INSERT INTO Zuordnung (MitID, ProNr, Istanteil, Plananteil) VALUES ('106', '31', 0.65, 0.1);

SELECT \* FROM Zuordnung WHERE MitID = 106;

**3.6**

INSERT INTO Projekt(ProNr, ProName, ProOrt, ProBeschreibung, ProAufwand, LeiterID) VALUES ('47', 'Oracle-Prakt', 'Dresden', NULL, 0, NULL);

SELECT \* FROM Projekt WHERE ProNr=47;

**24.11.2016**

**5 Abragen**

**5.1**

SELECT m.Nachname, m.Vorname, m.Ort, p.ProNr, p.ProName FROM Mitarbeiter m, Projekt p WHERE p.LeiterID = m.MitID

SELECT m.Nachname, m.Vorname, m.Ort, p.ProNr, p.ProName FROM Mitarbeiter m INNER JOIN Projekt p ON m.MitID = p.LeiterID

**5.2**

SELECT p.ProNr, SUM(Istanteil) FROM Projekt p, Zuordnung z WHERE p.LeiterID = z.MitID GROUP BY p.ProNr

SELECT p.ProNr, SUM(Istanteil) FROM Projekt p INNER JOIN Zuordnung z ON p.LeiterID = z.MitID GROUP BY p.ProNr

**5.3**

SELECT m.Nachname, m.Vorname, m.Ort, ProAufwand, p.ProNr, p.ProName, p.ProAufwand FROM Zuordnung z INNER JOIN Mitarbeiter m ON m.MitID = z.MitID

INNER JOIN Projekt p ON p.ProNr = z.ProNr AND p.LeiterID = m.MitID WHERE ProAufwand>=3

SELECT m.Nachname, m.Vorname, m.Ort, ProAufwand, p.ProNr, p.ProName, p.ProAufwand FROM Zuordnung z, Mitarbeiter m, Projekt p WHERE m.MitID = z.MitID

AND p.ProNr = z.ProNr AND p.LeiterID = m.MitID AND ProAufwand>=3

**5.4**

**a)**

SELECT p.ProNr, p.ProName, SUM(Istanteil) FROM Projekt p LEFT OUTER JOIN Zuordnung z ON p.LeiterID = z.MitID GROUP BY p.ProNr, p.ProName

**b)**

SELECT p.ProNr, ProName, SUM(Istanteil) FROM Projekt p, Zuordnung z WHERE p.ProNr = z.ProNr GROUP BY p.ProNr, ProName

UNION

SELECT ProNr, ProName, 0 FROM Projekt WHERE ProNr NOT IN (SELECT ProNr FROM Zuordnung)

**5.5**

--5.5 RICHTIGE LÖSUNG

SELECT m.MitID, 1 - SUM(Plananteil) AS Reserve FROM Mitarbeiter m, Zuordnung z WHERE m.MitID = z.MitID GROUP BY m.MitID

UNION

SELECT m.MitID, 1 FROM Zuordnung z, Mitarbeiter m WHERE m.MitID NOT IN (SELECT z.MitID FROM Zuordnung z)

SELECT p.ProNr, p.ProName, z.MitID, (1 - SUM(Plananteil)) AS Reserve

FROM Projekt p RIGHT OUTER JOIN Zuordnung z ON p.LeiterID = z.MitID

GROUP BY p.ProNr, p.ProName, z.MitID

SELECT p.ProNr, p.ProName, z.MitID, (1 - SUM(Plananteil)) AS Reserve

FROM Projekt p, Zuordnung z WHERE p.LeiterID = z.MitID

GROUP BY p.ProNr, p.ProName, z.MitID

**17.11.2016**

**3. Aufgabenkomplex**

**Teil 1**

* 1. **Skalarwertfunktion**

CREATE FUNCTION AlterErmitteln (@datum datetime)

RETURNS decimal (28,0)

AS

BEGIN

DECLARE @alter decimal (28,0)

SELECT @alter = DATEDIFF(yy, @datum, GETDATE())

RETURN @alter

END

SELECT \*, dbo.AlterErmitteln(Gebdat) AS [Alter] FROM Mitarbeiter

DROP FUNCTION AlterErmitteln

**1.2**

CREATE FUNCTION ErmittleAuslastung(@ast float)

RETURNS @table TABLE (

MitID char(3),

Auslastung float

)

AS

BEGIN

INSERT INTO @table

SELECT MitID, SUM(Istanteil) AS [Auslastung] FROM Zuordnung

GROUP BY MitID HAVING SUM(Istanteil)>@ast

--WHERE (Istanteil > @ast) GROUP BY MitID

RETURN

END

DROP FUNCTION ErmittleAuslastung

SELECT \* FROM dbo.ErmittleAuslastung(1)

**1.1 Teil II Defaults**

**a)**

ALTER TABLE Mitarbeiter ADD CONSTRAINT MA\_ORT

DEFAULT 'Dresden' FOR Ort

**b)**

ALTER TABLE Mitarbeiter ADD CONSTRAINT MA\_Beruf

DEFAULT 'Dipl.-Ing.' FOR Beruf

**24.11.2016**

**Teil 2 Check**

**2.1**

**f)**

ALTER TABLE Mitarbeiter ADD CONSTRAINT CK\_Mitarbeiter\_Alter CHECK(dbo.AlterErmitteln (Gebdat)>18 AND dbo.AlterErmitteln (Gebdat)<60)

**g)**

ALTER TABLE Mitarbeiter ADD CONSTRAINT CK\_Mitarbeiter\_MitID CHECK(CONVERT(INT, MitID) BETWEEN 001 AND 999)

ALTER TABLE Mitarbeiter DROP CONSTRAINT CK\_Mitarbeiter\_MitID

**2.2**

ALTER TABLE Zuordnung ADD CONSTRAINT CK\_Zuordnung\_Istanteil CHECK(Istanteil <= 1.0)

**Teil 3 Sicherung der referentiellen Integrität auf deklarativem Wege**

**3.1**

SELECT \* FROM Mitarbeiter

SELECT \* FROM Projekt

ALTER TABLE Projekt ALTER COLUMN LeiterID CHAR(3)

ALTER TABLE Projekt

ADD CONSTRAINT co\_forkey

FOREIGN KEY(LeiterID) REFERENCES Mitarbeiter(MitID)

ALTER TABLE Projekt DROP CONSTRAINT co\_forkey

**h)**

INSERT INTO Mitarbeiter (MitID, Nachname, Vorname, Ort, Gebdat, Beruf, Telnr) VALUES ('215', 'Day-Lewis', 'Daniel', 'London', '29.04.1957', 'Praktikant', '00491746754686')

INSERT INTO Projekt (ProNr, ProName, ProOrt, ProBeschreibung, ProAufwand, LeiterID) VALUES ('48', 'Statistik', 'Dresden', '', '2', '215')

UPDATE Mitarbeiter SET MitID='216' WHERE Nachname = 'Day-Lewis'

**i)**

DELETE FROM Mitarbeiter WHERE MitID = '216' --GEHT NICHT UND MUSS AUCH NICHT GEHEN

**j)**

UPDATE Projekt SET LeiterID='210' WHERE ProNr = '48' --GEHT

DELETE FROM Mitarbeiter WHERE MitID = '210' --GEHT NICHT

**k)**

INSERT INTO Projekt (ProNr, ProName, ProOrt, ProBeschreibung, ProAufwand, LeiterID) VALUES ('55', 'Statistik', 'Dresden', '', '2', '300')

* **GEHT NICHT**

**l)**

INSERT INTO Projekt (ProNr, ProName, ProOrt, ProBeschreibung, ProAufwand, LeiterID) VALUES ('55', 'Integritätssicherung', 'Dresden', '', '2', NULL)

**--GEHT**

**TEIL IV** Sicherung der referentiellen bzw. semantischen Integrität-Prozedurale Lösung

**4 Stored procedure mit Cursor**

**a)**

CREATE PROCEDURE MA\_Daten (@pnr INT)

AS

BEGIN

--DECLARE @pname VARCHAR(8), @paufwand INT

--SELECT @pname = ProName, @paufwand = ProAufwand FROM Projekt WHERE ProNr=@pnr

SELECT ProName, ProAufwand, MitID, Nachname, Vorname FROM Projekt, Mitarbeiter WHERE ProNr = @pnr AND LeiterID = MitID

RETURN

END

--DROP PROCEDURE MA\_Daten

MA\_Daten 48

**b)**

ALTER PROCEDURE MA\_Daten (@pnr INT)

AS

BEGIN

--DECLARE @pname VARCHAR(8), @paufwand INT

--SELECT @pname = ProName, @paufwand = ProAufwand FROM Projekt WHERE ProNr=@pnr

DECLARE @mitid CHAR(3)

DECLARE @nachname VARCHAR(20), @vorname VARCHAR(20), @beruf VARCHAR(15), @msg VARCHAR(120)

DECLARE @plananteil FLOAT, @istanteil FLOAT, @abweichung FLOAT

SELECT ProName, ProAufwand, MitID, Nachname, Vorname FROM Projekt, Mitarbeiter WHERE ProNr = @pnr AND LeiterID = MitID

------------------------b-------------------------------------------------------

IF ((SELECT COUNT(\*) FROM Projekt WHERE ProNr = @pnr) = 0)

BEGIN

PRINT 'Das Projekt gibt es im Unternehmen nicht.'

RETURN

END

DECLARE madaten CURSOR FOR

SELECT m.MitID, m.Nachname, m.Vorname, m.Beruf, z.Plananteil, z.Istanteil, (Plananteil - Istanteil)

FROM Mitarbeiter m, Zuordnung z

WHERE z.ProNr = @pnr AND z.MitID = m.MitID

--SELECT @msg = @mitid + '' + @nachname + '' + @vorname + ''

-- + @beruf + '' + @plananteil + ''+ @istanteil+ '' +@abweichung + ''

--PRINT @msg

--PRINT '-------------------------------------------------------------------------------------------------------'

OPEN madaten

FETCH madaten INTO @mitid, @nachname, @vorname, @beruf, @plananteil, @istanteil, @abweichung

IF (@@FETCH\_STATUS = -1)

BEGIN

PRINT 'Das Projekt gibt es nicht'

CLOSE madaten

DEALLOCATE madaten

RETURN

END

WHILE (@@FETCH\_STATUS = 0)

BEGIN

--SELECT @abweichung = @plananteil - @istanteil

SELECT @msg = @mitid + ' ' + @nachname + ' ' + @vorname + ' ' + @beruf + ' '

+ CONVERT(VARCHAR(12), @plananteil) + ' '

+ CONVERT(VARCHAR(12), @istanteil) + ' ' + CONVERT(VARCHAR(12), @abweichung) + ' '

PRINT '-----------------------------------------------------------------------------------------------------------------'

PRINT @msg

FETCH NEXT FROM madaten INTO @mitid, @nachname, @vorname, @beruf, @plananteil, @istanteil, @abweichung

--FETCH madaten INTO @plananteil, @istanteil, @abweichung

END

PRINT '--------------------------------------------------------------------------------------------------'

CLOSE madaten

DEALLOCATE madaten

--RETURN

END

--MA\_Daten 31

--DROP PROCEDURE MA\_Daten

**1.Aufgabenkomplex**

**Einfügen, Ändern von Daten**

**3.5**

SELECT \* FROM Zuordnung WHERE MitID = 106;

DELETE FROM Zuordnung WHERE MitID = '106' AND Istanteil = 0.65

INSERT INTO Zuordnung (MitID, ProNr, Istanteil, Plananteil)

VALUES ('106', '31', 0.65, (SELECT Plananteil = 1.0-SUM(Plananteil) FROM Zuordnung WHERE MitID = '106') )

**AUFGABENKOPLEX 4**

--SELECT \* INTO Zuordnung\_Kopie FROM Zuordnung

-- CREATE USER Oksana FOR LOGIN [smb\s74011] WITH DEFAULT\_SCHEMA = extern

--Aufgabenkomplex 4

-- 2.5

--SELECT \* FROM dbo.Mitarbeiter

-- 2.1

--use iw15s74011

--2.9

BEGIN TRANSACTION

SELECT \* FROM dbo.Mitarbeiter with (HOLDLOCK)

--COMMIT TRANSACTION

ROLLBACK TRANSACTION

--2.14

CREATE TABLE Mitarbeiter (MitID CHAR(3) NOT NULL,

Nachname VARCHAR(20) NOT NULL,

Vorname VARCHAR(20),

Ort VARCHAR(30) NOT NULL,

Gebdat DATE NOT NULL,

Beruf VARCHAR(15) NOT NULL,

Telnr VARCHAR(20),

PRIMARY KEY(MitID))

INSERT INTO Mitarbeiter VALUES ('555', 'Squarepants', 'Spongebob', 'Dresden', '02.04.1992', 'Master', '00471746353598' )

SELECT \* FROM Mitarbeiter

--2.17

DROP TABLE Mitarbeiter

USE iw15s73889

**OKSANA**

--2. Aufgaben zu Zugriffsrechten  
  
--2.   
  
CREATE USER merko FOR LOGIN [smb\s73889]  
    WITH DEFAULT\_SCHEMA = extern  
--3.  
CREATE SCHEMA extern AUTHORIZATION merko    
  
--6.  
  
GRANT SELECT ON Mitarbeiter to merko  
  
--8.  
  
sp\_who  
sp\_helprotect  
  
--10.  
  
UPDATE Mitarbeiter SET Nachname='Surkan' WHERE Nachname='Tsurkan'  
  
--geht nicht, es bleibt 'Abfrage wird ausgeführt' stehen, nachdem ROLLBACK TRANSACTION vom anderen user ausgeführt wird, geht das  
  
--11.   
  
sp\_lock  
  
--13.   
  
GRANT CREATE TABLE TO merko  
  
--15.  
select \* from extern.Mitarbeiter  
  
--18.  
  
REVOKE CREATE TABLE FROM merko  
REVOKE SELECT ON Mitarbeiter FROM merko  
  
DROP SCHEMA extern  
  
DROP USER merko

**01.12.2016**

**Teil IV**

**4.1**

**Die Werte werden erhöht und 1 wird überschritten**

ALTER TRIGGER plananteil\_pruef ON Zuordnung

FOR UPDATE AS

IF UPDATE(Plananteil)

BEGIN

DECLARE @wert\_pa FLOAT

DECLARE pa\_cursor CURSOR FOR

(SELECT SUM(Plananteil) FROM INSERTED

WHERE MitID IN (SELECT MitID FROM INSERTED)

GROUP BY MitID)

OPEN pa\_cursor

FETCH pa\_cursor INTO @wert\_pa

IF (@@FETCH\_STATUS = -1)

BEGIN

IF (@wert\_pa > 1)

BEGIN

ROLLBACK TRANSACTION

PRINT 'Achtung Summe der Plananteil würde 1 überschreiten. Änderung wurde ausgeführt.'

--BREAK

--CLOSE pa\_cursor

--DEALLOCATE pa\_cursor

END

END

WHILE(@@FETCH\_STATUS = 0)

BEGIN

FETCH pa\_cursor INTO @wert\_pa

END

CLOSE pa\_cursor

DEALLOCATE pa\_cursor

PRINT 'Die Änderung der Plananteile wurde ausgeführt.'

END

drop trigger plananteil\_pruef

UPDATE Zuordnung SET Plananteil = Plananteil - 0.3 WHERE ProNr = 31

SELECT Plananteil FROM Zuordnung WHERE ProNr = 31

SELECT \* FROM Zuordnung WHERE ProNr = 44

SELECT \* FROM Zuordnung WHERE MitID = '105'

UPDATE Zuordnung SET Plananteil = Plananteil + 0.4 WHERE MitID = '105'

**4.1**

LÖSUNG

ALTER TRIGGER plananteil\_pruef ON Zuordnung

FOR UPDATE AS

IF UPDATE(Plananteil)

BEGIN

DECLARE @wert\_pa FLOAT

DECLARE pa\_cursor CURSOR FOR

(SELECT SUM(Plananteil) FROM INSERTED

WHERE MitID IN (SELECT MitID FROM INSERTED)

GROUP BY MitID)

OPEN pa\_cursor

FETCH pa\_cursor INTO @wert\_pa

WHILE(@@FETCH\_STATUS = 0)

BEGIN

IF (@wert\_pa > 1)

BEGIN

ROLLBACK TRANSACTION

PRINT 'Achtung Summe der Plananteil würde 1 überschreiten. Änderung wurde nicht ausgeführt.'

END

FETCH NEXT FROM pa\_cursor INTO @wert\_pa

END

CLOSE pa\_cursor

DEALLOCATE pa\_cursor

PRINT 'Die Änderung der Plananteile wurde ausgeführt.'

END

drop trigger plananteil\_pruef

UPDATE Zuordnung SET Plananteil = Plananteil + 0.5 WHERE ProNr = 31

SELECT Plananteil FROM Zuordnung WHERE ProNr = 31

SELECT \* FROM Zuordnung WHERE MitID = '105'

UPDATE Zuordnung SET Plananteil = Plananteil + 0.4 WHERE MitID = '105'

**4.2**

ALTER TRIGGER plananteil\_del ON Zuordnung

FOR UPDATE, DELETE AS

BEGIN

DECLARE @pnr INT

DECLARE pnr\_cursor CURSOR FOR

SELECT ProNr FROM DELETED WHERE ProNr NOT IN (SELECT ProNr FROM INSERTED)

OPEN pnr\_cursor

FETCH pnr\_cursor INTO @pnr

IF(@@FETCH\_STATUS = 0)

BEGIN

UPDATE Projekt SET ProBeschreibung = 'erledigt' WHERE ProNr = @pnr

END

FETCH pnr\_cursor INTO @pnr

CLOSE pnr\_cursor

DEALLOCATE pnr\_cursor

END

SELECT \* FROM Projekt

SELECT \* FROM Zuordnung

INSERT INTO Projekt(ProNr, ProName, ProOrt, ProBeschreibung, ProAufwand, LeiterID)

VALUES ('63', 'MS-SQL-Prakt', 'Dresden', 'NULL', '3', '210');

INSERT INTO Zuordnung(MitID, ProNr, Istanteil, Plananteil)

VALUES ('210', '63', 0.5, 0.5)

UPDATE Projekt SET ProBeschreibung = NULL WHERE LeiterID=210

DELETE FROM Zuordnung WHERE ProNr = 63

**5.1**

/\*

CREATE TABLE BProtokoll (MitID CHAR(3) NOT NULL,

Nachname VARCHAR(16) NOT NULL,

Zeit DATETIME NOT NULL,

Beruf\_alt VARCHAR(15) NOT NULL,

Beruf\_neu VARCHAR(15) NOT NULL,

PRIMARY KEY (MitID, Zeit)

) \*/

--SELECT \* FROM BProtokoll

--DROP TABLE BProtokoll

**5.2**

ALTER TRIGGER Protokoll ON Mitarbeiter

FOR UPDATE AS

IF UPDATE(Beruf)

BEGIN

DECLARE @beruf\_alt VARCHAR(15)

DECLARE @beruf\_neu VARCHAR(15)

DECLARE @id VARCHAR(3)

DECLARE cursor1 CURSOR FOR

SELECT i.MitID, d.Beruf, i.Beruf FROM INSERTED i, DELETED d

WHERE i.MitID = d.MItID

PRINT 'TEST1'

OPEN cursor1

FETCH cursor1 INTO @id, @beruf\_alt, @beruf\_neu

PRINT @beruf\_alt

PRINT @beruf\_neu

WHILE(@@FETCH\_STATUS = 0)

BEGIN

INSERT INTO BProtokoll VALUES

( @id, (user\_name()), (getdate()), @beruf\_alt, @beruf\_neu)

FETCH cursor1 INTO @id, @beruf\_alt, @beruf\_neu

END

CLOSE cursor1

DEALLOCATE cursor1

END

SELECT \* FROM BProtokoll

UPDATE Mitarbeiter SET Beruf ='Student' WHERE Nachname = 'Merko' AND MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Akteur' WHERE Nachname = 'Day-Lewis' AND MitID = '215'

SELECT \* FROM Mitarbeiter

UPDATE Mitarbeiter SET Beruf = 'BAUER' WHERE Ort = 'Dresden'

**5.3**

SELECT \* FROM BProtokoll

SELECT \* FROM Mitarbeiter WHERE MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Praktikant' WHERE Nachname = 'Merko' AND MitID = '210'

**5.5**

SELECT \* FROM Mitarbeiter WHERE MitID = '215'

UPDATE Mitarbeiter SET Beruf ='Anwalt' WHERE Nachname = 'Day-Lewis' AND MitID = '215'

ALTER TRIGGER Lösch\_MA ON Mitarbeiter

FOR DELETE AS

BEGIN

DELETE FROM BProtokoll WHERE MitID IN (SELECT DISTINCT MitID FROM DELETED)

END

SELECT \* FROM Mitarbeiter WHERE MitID = '215'

SELECT \* FROM Mitarbeiter WHERE MitID = '210'

SELECT \* FROM BProtokoll

DELETE FROM Mitarbeiter WHERE MitID = '215'

INSERT INTO Mitarbeiter (MitID, Nachname, Vorname, Ort, Gebdat, Beruf, Telnr)

VALUES ('215', 'Day-Lewis', 'Daniel', 'London', '29.04.1957', 'Praktikant', '00491746754686')

SELECT \* FROM Mitarbeiter WHERE MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Student' WHERE Nachname = 'Merko' AND MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Akteur' WHERE Nachname = 'Day-Lewis' AND MitID = '215'

**5.6**

SELECT \* FROM BProtokoll

ALTER TABLE BProtokoll ADD ID INT IDENTITY(1,1)

ALTER TABLE BProtokoll DROP COLUMN ID

ALTER TRIGGER Protokoll ON Mitarbeiter

FOR UPDATE AS

IF UPDATE(Beruf)

BEGIN

DECLARE @beruf\_alt VARCHAR(15)

DECLARE @beruf\_neu VARCHAR(15)

SELECT @beruf\_alt = (SELECT Beruf FROM DELETED)

SELECT @beruf\_neu = (SELECT Beruf FROM INSERTED)

INSERT INTO BProtokoll(MitID, Nachname, Zeit, Beruf\_alt, Beruf\_neu)

VALUES

((SELECT MitID FROM INSERTED), (SELECT Nachname FROM INSERTED), (getdate()), @beruf\_alt, @beruf\_neu)

END

UPDATE Mitarbeiter SET Beruf ='Student' WHERE Nachname = 'Merko' AND MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Akteur' WHERE Nachname = 'Day-Lewis' AND MitID = '215'

**5.8 und 5.9**

/\*CREATE TABLE BProtokoll2 (DSNo INT,

MitID CHAR(3) NOT NULL,

Nachname VARCHAR(16) NOT NULL,

Zeit DATETIME NOT NULL,

Beruf\_alt VARCHAR(15) NOT NULL,

Beruf\_neu VARCHAR(15) NOT NULL,

PRIMARY KEY (DSNo, MitID, Zeit)

) \*/

ALTER TRIGGER Protokoll2 ON Mitarbeiter

FOR UPDATE AS

IF UPDATE(Beruf)

BEGIN

DECLARE @beruf\_alt VARCHAR(15)

DECLARE @beruf\_neu VARCHAR(15)

DECLARE @zaehler INT

SELECT @beruf\_alt = (SELECT Beruf FROM DELETED)

SELECT @beruf\_neu = (SELECT Beruf FROM INSERTED)

--SELECT @zaehler = (SELECT DSNo from BProtokoll2)

IF ((SELECT COUNT(\*) FROM BProtokoll2) = 0 )

BEGIN

SELECT @zaehler = 1

END

ELSE

BEGIN

SELECT @zaehler = ((SELECT MAX(DSNo) FROM BProtokoll2) +1 )

END

INSERT INTO BProtokoll2 (DSNo, MitID, Nachname, Zeit, Beruf\_alt, Beruf\_neu)

VALUES

(@zaehler, (SELECT MitID FROM INSERTED), (SELECT Nachname FROM INSERTED), (getdate()), @beruf\_alt, @beruf\_neu)

END

SELECT \* FROM BProtokoll2

DELETE FROM BProtokoll2 WHERE MitID = '215' or MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Student' WHERE Nachname = 'Merko' AND MitID = '210'

UPDATE Mitarbeiter SET Beruf ='Akteur' WHERE Nachname = 'Day-Lewis' AND MitID = '215'

**5.10**

ALTER TRIGGER Protokoll2 ON Mitarbeiter

FOR UPDATE AS

IF UPDATE(Beruf)

BEGIN

DECLARE @beruf\_alt VARCHAR(15)

DECLARE @beruf\_neu VARCHAR(15)

DECLARE @zaehler INT

DECLARE @id CHAR(3)

DECLARE @nachname CHAR(15)

DECLARE pkurs CURSOR FOR

SELECT DSNo, MitID, Nachname, Zeit, Beruf\_alt, Beruf\_neu FROM BProtokoll2

FETCH pkurs INTO @zaehler, @id, user\_name(), getdate(), @beruf\_alt, @beruf\_neu

WHILE(@@FETCH\_STATUS = -1)

BEGIN

SELECT @zaehler = 1

END

IF (@@FETCH\_STATUS = 0)

BEGIN

SELECT @beruf\_alt = (SELECT Beruf FROM DELETED)

SELECT @beruf\_neu = (SELECT Beruf FROM INSERTED)

SELECT @zaehler = ((SELECT MAX(DSNo) FROM BProtokoll2) +1 )

END

INSERT INTO BProtokoll2 (DSNo, MitID, Nachname, Zeit, Beruf\_alt, Beruf\_neu)

VALUES

(@zaehler, (SELECT MitID FROM INSERTED), (SELECT Nachname FROM INSERTED), (getdate()), @beruf\_alt, @beruf\_neu)

END

FETCH pkurs INTO @zaehler, @id, user\_name(), getdate(), @beruf\_alt, @beruf\_neu

CLOSE pkurs

DEALLOCATE pkurs

SELECT \* FROM BProtokoll2

DELETE FROM BProtokoll2 WHERE MitID = '215'

UPDATE Mitarbeiter SET Beruf ='Student' WHERE MitID = '210' OR MitID = '215'

UPDATE Mitarbeiter SET Beruf ='Akteur' WHERE Nachname = 'Day-Lewis' AND MitID = '215'

ALTER TRIGGER Protokoll2 ON Mitarbeiter

FOR UPDATE AS

IF UPDATE(Beruf)

BEGIN

DECLARE @beruf\_alt VARCHAR(15)

DECLARE @beruf\_neu VARCHAR(15)

DECLARE @id CHAR(3)

DECLARE pkurs CURSOR FOR

SELECT m.MitID, d.Beruf, i.Beruf FROM Mitarbeiter m, INSERTED i, DELETED d

WHERE m.MitID = i.MitID AND m.MitID = d.MitID AND d.MitID = i.MitID

FETCH pkurs INTO @id, @beruf\_alt, @beruf\_neu

WHILE (@@FETCH\_STATUS = 1)

BEGIN

INSERT INTO BProtokoll2 (DSNo, MitID, Nachname, Zeit, Beruf\_alt, Beruf\_neu)

VALUES

(( SELECT MAX(DSNo) FROM BProtokoll2) +1, @id, user\_name(), getdate(), @beruf\_alt, @beruf\_neu )

FETCH pkurs INTO @id, @beruf\_alt, @beruf\_neu

END

CLOSE pkurs

DEALLOCATE pkurs

END

SELECT \* FROM BProtokoll2

DELETE FROM BProtokoll2 WHERE MitID = '215'

UPDATE Mitarbeiter SET Beruf ='Student' WHERE MitID = '210' OR MitID = '215'

UPDATE Mitarbeiter SET Beruf ='Akteur' WHERE Nachname = 'Day-Lewis' AND MitID = '215'