**Datenbankdokumentation**

**Tabellen**

ER-Diagramm als PDF und SVG beiliegend.

**Abwesenheit:**

tr\_id INT(10) Primärschlüssel, Fremdschlüssel (Trainingseinheit)

p\_id INT(10) Primärschlüssel, Fremdschlüssel (Person)

**Aufstellung:**

s\_id INT(10) Primärschlüssel, Fremdschlüssel (Spiel)

p\_id INT(10) Primärschlüssel, Fremdschlüssel (Person)

**Mannschaft:**

m\_id INT(10) Primärschlüssel

name VARCHAR(50) UNIQUE

**Mannschaft\_Turnier\_Sparte**

m\_id INT(10) Primärschlüssel, Fremdschlüssel (Mannschaft)

tu\_id INT(10) Primärschlüssel, Fremdschlüssel (Turnier)

sparte\_id INT(10) Primärschlüssel, Fremdschlüssel (Sparte)

**Person:**

p\_id INT(10) Primärschlüssel

name VARCHAR(50)

v\_name VARCHAR(50)

geb\_datum DATE

groesse TINYINT(4) Nullable

bild VARCHAR(100)

betreuer TINYINT(1)

tel VARCHAR(50) Nullable

username VARCHAR(105) UNIQUE

password VARCHAR(60)

**Sparte:**

sparte\_id INT(10) Primärschlüssel

name VARCHAR(50) UNIQUE

**Spiel:**

s\_id INT(10) Primärschlüssel

ort VARCHAR(200)

heim INT(10) Fremdschlüssel (Mannschaft)

auswaerts INT(10) Fremdschlüssel (Mannschaft)

h\_tore TINYINT(4)

a\_tore TINYINT(4)

stat\_id INT(10) Fremdschlüssel (Status)

zeit DATETIME

tu\_id INT(10) Fremdschlüssel (Turnier)

sparte\_id INT(10) Fremdschlüssel (Sparte)

**Status:**

stat\_id INT(10) Primärschlüssel

status VARCHAR(30) UNIQUE

**Teilnehmer\_TG:**

tg\_id INT(10) Primärschlüssel, Fremdschlüssel (Trainingsgruppe)

p\_id INT(10) Primärschlüssel, Fremdschlüssel (Person)

**Trainingseinheit:**

tr\_id INT(10) Primärschlüssel

name VARCHAR(50)

ort VARCHAR(200)

zeit DATETIME

trainer INT(10) Fremdschlüssel (Person)

tg\_id INT(10) Fremdschlüssel (Trainingsgruppe)

**Trainingsgruppe:**

tg\_id INT(10) Primärschlüssel

name VARCHAR(50) UNIQUE

**Turnier:**

tu\_id INT(10) Primärschlüssel

name VARCHAR(50) UNIQUE

**Turnier\_Sparte**

tu\_id INT(10) Primärschlüssel, Fremdschlüssel (Turnier)

sparte\_id INT(10) Primärschlüssel, Fremdschlüssel (Sparte)

gewinner INT(10) Fremdschlüssel (Mannschaft)

**Trigger:**

**AutoDeleteAbwesenheit**

Der Trigger löscht nach einem DELETE in der Tabelle Teilnehmer\_tg alle Einträge in der Tabelle Abwesenheit, die die gleiche ‘p\_id‘ und die passenden ‘tr\_id‘ besitzen.

SQL-Befehl:

CREATE TRIGGER `AutoDeleteAbwesenheit` AFTER DELETE ON `teilnehmer\_tg` FOR EACH ROW BEGIN

DELETE FROM abwesenheit

WHERE abwesenheit.p\_id = OLD.p\_id AND

abwesenheit.tr\_id IN (SELECT trainingseinheit.tr\_id

FROM trainingseinheit

WHERE trainingseinheit.tg\_id = OLD.tg\_id);

END

**AutoInsertAbwesenheit**

Der Trigger fügt nach einem INSERT in der Tabelle Teilnehmer\_tg in die Tabelle Abwesenheit Einträge ein, die gleiche ‘p\_id‘, die passenden ‘tr\_id‘ besitzen.

SQL-Befehl:

CREATE TRIGGER `AutoInsertAbwesenheit` AFTER INSERT ON `teilnehmer\_tg` FOR EACH ROW BEGIN

INSERT INTO abwesenheit

SELECT trainingseinheit.tr\_id, NEW.p\_id

FROM trainingseinheit

WHERE trainingseinheit.tg\_id = NEW.tg\_id AND trainingseinheit.zeit > NOW();

END

**AutoInsertFreundschaft**

Der Trigger fügt nach einem INSERT in der Tabelle Mannschaft für die neue Mannschaft in die Tabelle Mannschaft\_Turnier\_Sparte Einträge mit dem Turnier ‘Freundschaftsspiel‘ und jeder Sparte ein.

SQL-Befehl:

CREATE TRIGGER `AutoInsertFreundschaft` AFTER INSERT ON `mannschaft` FOR EACH ROW BEGIN

INSERT INTO mannschaft\_turnier\_sparte

SELECT New.m\_id, turnier.tu\_id, sparte.sparte\_id

FROM sparte,turnier

WHERE turnier.name ='Freundschaftsspiel';

END

**BefuellenAbwesenheit**

Der Trigger befüllt die Tabelle Abwesenheit, wenn eine Trainingseinheit erstellt wird.

SQL-Befehl:

CREATE TRIGGER `BefuellenAbwesenheit` AFTER INSERT ON `trainingseinheit` FOR EACH ROW BEGIN

INSERT INTO abwesenheit

SELECT NEW.tr\_id, teilnehmer\_tg.p\_id

FROM teilnehmer\_tg

WHERE teilnehmer\_tg.tg\_id = NEW.tg\_id;

END

**UpdateTraining**

Der Trigger aktualisiert die Tabelle Abwesenheit, wenn bei einer Trainingseinheit die Trainingsgruppe verändert wird.

SQL-Befehl:

CREATE TRIGGER `UpdateTraining` AFTER UPDATE ON `trainingseinheit` FOR EACH ROW BEGIN

IF OLD.tg\_id <> NEW.tg\_id THEN

DELETE FROM abwesenheit WHERE abwesenheit.tr\_id = NEW.tr\_id;

INSERT INTO abwesenheit

SELECT NEW.tr\_id, teilnehmer\_tg.p\_id

FROM teilnehmer\_tg

WHERE teilnehmer\_tg.tg\_id = NEW.tg\_id;

END IF;

END

Weiterhin existieren zu jeder Tabelle, in der VARCHARs eingefügt werden, Trigger, die vor jedem INSERT und UPDATE die Einträge in VARCHAR-Spalten von voran- oder nachgestellte Leerzeichen bereinigen. Dazu als Beispiel:

**TrimMannschaftInsert**

SQL-Befehl:

CREATE TRIGGER `TrimMannschaftInsert` BEFORE INSERT ON `mannschaft` FOR EACH ROW BEGIN

SET NEW.name = TRIM(NEW.name);

END