Datamining

Franz Fiala

Ja, wenn die Datenbank einmal fertig ist, dann hat man fast schon gewonnen, denn die eingangs beschrieben Auswertung ist nur der Endpunkt eines mühsamen Weges. Aber wie kommt man zu diesen Daten ohne sie mühsam abtippen zu müssen?

Der ÖFB publiziert auf seiner Statistik-Seite *http://www.oefb.at/statistik-pid623#el15406* in 12 PDF-Dateien alle Spiele der österreichischen Nationalmannschaft seit 1902:


LÄNDERSPIELSTATISTIK VON 1902 BIS HEUTE

- Länderspiele von 1902 - August 1923 (Länderspiele Nr. 1-80)
- Länderspiele von September 1923 - April 1934 (Länderspiele Nr. 81-162)
- Länderspiele von April 1934 - Mai 1952 (Länderspiele Nr. 163-236)
- Länderspiele von Juni 1952 - November 1960 (Länderspiele Nr. 237-300)
- Länderspiele von Dezember 1960 bis April 1966 (Länderspiele Nr. 301-336)
- Länderspiele von Mai 1966 - Mai 1982 (Länderspiele Nr. 337-455)
- Länderspiele von Juni 1982 - September 1988 (Länderspiele Nr. 456-500)
- Länderspiele von Oktober 1988 - Februar 2000 (Länderspiele Nr. 501-599)
- Länderspiele von März 2000 - Oktober 2003 (Länderspiele Nr. 600-631)
- Länderspiele von März 2004 - November 2007 (Länderspiele Nr. 632-668)
- Länderspiele 2008 - 2010 (Länderspiele Nr. 669-697)
- Länderspiele 2011 - 2012 (Länderspiele Nr. 698-716)
- Länderspiele seit 2013

Liste der Länderspiele der österreichischen Nationalmannschaft, in 12 PDF-Dokumenten

Die Spiele sind fortlaufend nummeriert. Das erste Spiel gegen Ungarn vom 12.10.1902 hat die Nummer 1 und das letzte gegen Uruguay vom 5.3.2014 hat die Nummer 727. Das letzte Spiel in der ÖFB-Publikation ist das Spiel 716 gegen die Elfenbeinküste vom 14.11.2012.

Der Originaltext in der PDF-Datei ist so wie in den beiden Bildern rechts.

Man sieht an der Darstellung dieser beiden Spiele, dass sich die Zahl der Angaben zum Spiel im Laufe der vielen Jahre sehr verändert hat. Zwischen dem Minimum des ersten Spiels und der ganzen Seite des letzten Spiels ist ein großer formaler Unterschied.

Für Leser ist das unbedeutend aber für ein Programm, das diese Angaben in eine Tabellenstruktur umwandeln soll, ist das eine Herausforderung.

Zuerst müssen die 12 PDF-Dateien zu einem einzigen Dokument verbunden werden. Dazu eignet sich ***Acrobat*** oder das preisgünstigere ***Foxit***.

Die so gewonnene PDF-Datei `alles-original.pdf` wird mit dem OCR-Programm ***Omnipage*** in Text umgewandelt. Der Anfang dieser neuen Datei (`alles-original.txt`) schaut so aus wie im Bild rechts unten.

Während die Datei `alles-original.pdf` noch ein Bild war, arbeiten wir jetzt mit reinem Text. Natürlich mit kleinen Tippfehlern, etwa sieht man, dass gleich in der ersten Zeile „Osterreich“ „O“ statt „Ö“ steht. Diese Fehler wiederholen sich im Dokument und werden – wie viele andere Kleinigkeiten durch Suchen und Ersetzen beseitigt.

Grob kann man in diesen Angaben zu den Spielen die Datenbankstruktur erkennen. Jedes Spiel besteht eigentlich aus drei tabellarischen Strukturen:

- aus den Angaben zum Spiel (Datum, Ergebnis, Gegner…),
- den Spielern, die an dem Spiel teilnehmen, den Vereinen denen sie angehören, ergänzt

Österreich : Ungarn
5:0
Freundschaftsspiel
1. Länderspiel, Wien (WAC-Platz), 12. 10. 1902
Schiedsrichter: Shires, AUT

Format 1. Länderspiel, 1902

Nauss (WAC); **Eipeldauer** (Vienna), **Wachuda** (WAC), **Hüttl** (Cricketer), **Blässy** (Graphia), **Mössmer** (DJM Währing), **Wiesner** (WFC 1898), **Huber** (WAC), **Schrammel** (WAC), **Studnicka** (WAC), **Taurer** (WAC)

Tore: Huber, Taurer, Studnicka (3x)

716. Länderspiel

Format 716. Länderspiel 2012

Österreich – Elfenbeinküste 0:3 (0:1)
Freundschaftliches Länderspiel, 14. November 2012
Linz, Linzer Stadion, 13.832 Zuschauer, SR Kralovec (CZE)

Österreich spielte mit:
(GK 24) Heinz Lindner/FK Austria Wien – (2) György **Garics**/FC Bologna, (3) Aleksandar **Dragovic**/FC Basel, (4) Emanuel **Pogatetz**/VfL Wolfsburg, (13) Markus **Suttner**/FK Austria Wien – (7) Marko **Arnautovic**/SV Werder Bremen, (18) Christoph **Leitgeb**/FC RB Salzburg, (8) David **Alaba**/FC Bayern München, (16) Jakob **Jantscher**/Dynamo Moskau – (6) Andreas **Ivanschitz**/1.FSV Mainz 05 – (21) Marc **Janko**/Trabzonspor

Austausch:
(17) Florian **Klein**/FC RB Salzburg für Garics (46.), (15) Sebastian **Prödl**/SV Werder Bremen für Pogatetz (46.), (14) Julian **Baumgartlinger**/1.FSV Mainz 05 für Leitgeb (46.), (19) Veli **Kavlak**/Besiktas JK für Alaba (59.), (9) Andreas **Weimann**/Aston Villa FC für Ivanschitz (64.), (11) Martin **Harnik**/VfB Stuttgart für Jantscher (76.)

Teamchef : Marcel **Koller**

Elfenbeinküste spielte mit:

Boubacar **Barry**/KSC Lokeren, Igor **Lolo**/Kuban Krasnodar, Kolo **Touré**/Manchester City, Sol **Bamba**/Trabzonspor, Arthur **Boka**/VfB Stuttgart – **Romaric**/Real Saragossa, Arouna **Kone**/Wigan Athletic, Cheick **Tioté**/Newcastle United, Didier **Ya Konan**/Hannover 96, Max **Gradel**/AS Saint-Étienne – Wilfried **Bony**/Vitesse Arnheim

Austausch:
Ismael **Traore**/Stade Brestois für Bamba (46.), Lacina **Traoré**/Anzhi Makhachkala für Bony (46.), Abdul **Razak**/Manchester City für Tiote (53.), Didier **Drogba**/Shanghai Shenhua für A. Kone (59.), Yaya **Touré**/Manchester City für Romaric, Salomon **Kalou**/Lille OSC für Ya Konan

Teamchef : Sabri **Lamouchi**

Tore:
0:1 Ya Konan (44.)
0:2 Drogba (52.)
0:3 L. Traore (76.)

Alles.txt: reiner Text nach Konversion der Datei alles.pdf, mit typischen OCR-Fehlern.

Osterreich : Ungarn¶
5:0¶
Freundschaftsspiel¶
1. Länderspiel, Wien (WAC-Platz), 12. 10. 1902 Schiedsrichter: Shires, AUT¶
Nauss (WAC); Eipeldauer (Vienna), Wachuda (VVAC), Hüttl (Cricketer), Blässy (Graphia), Mössmer (DJM VVahring), Wiesner (WFC 1898), Huber (VVAC), Schrammel (WAC), Studnicka (VVAC), Taurer (WAC)¶
Tore: Huber, Taurer, Studnicka (3x)¶
Ungarn : Österreich¶
3:2¶
(1:0)¶
Freundschaftsspiel¶
2. Länderspiel, Budapest, 11.06. 1903 Schiedsrichter: Jolland, H¶
Wagner (Cricketer); Leuthe (WAC), Dettelmaier (WAC), Schrammel (WAC), Stürmer (WAC), Dick (Deutscher Sportverein), Fischer (WAC), Schulz (WAC), Pulchert (Vienna), Studnicka (WAC), Taurer (WAC)¶
Tore: Pulchert, Studnicka¶

durch die Spielernummer und die Spielzeit (von Minute, bis Minute), die Aufstellung

- Die Auswechslungen, wobei jeweils angegeben ist, welcher Spieler gegen welchen anderes Spieler in welcher Minute getauscht wurde und schließlich
- den Toren der Begegnung, ergänzt durch die Minute, den Schützen und wer das Tor geschossen hat.

In einem ersten Schritt werden die Angaben zum Spiel in alle diese Einzelteile zerlegt und die **Aufstellung** und **Tore** noch als eine Zeichenkette beisammen gelassen aber die Angaben zum Spiel bereits in die Felder **Datum**, **Gegner**, **Ergebnis**, **Halbzeitergebnis**, **Schiedsrichter**, **Zuschauerzahl**, **Spielort** usw. aufgeteilt.

Die **Aufstellung** und die **Tore** bleiben zunächst als Text in einem genügend großen Textfeld. 255 Zeichen (die Vorgabe *„Kurzer Text“* in Microsoft Access) genügt nicht, man muss *„Langer Text“* einstellen, sonst werden bei einigen Spielen Textteile abgeschnitten. In diesem ersten Schritt entsteht also die einzige Tabelle **Spiel**.

Dazu muss aber die Textdatei gründlich umformatiert werden, damit ein lesendes Programm einfach erkennen kann, wann ein Spiel beginnt, wann es endet und welche Abschnitte daraus in welches Feld kommen sollen.

Das folgende Beispiel zeigt am Spiel 716 gegen die Elfenbeinküste, das auf der vorigen Seite auch im Original zu sehen ist, wie dieser maschinenlesbare Text schließlich ausschaut. Zur Verbesserung der Lesbarkeit wurden die Feldnamen fett gedruckt.

####
***#Begegnung#**Österreich:Elfenbeinküste*
***#Spiel#**716*
***#Ergebnis#**0:3*
***#Pause#**0:1*
***#Art#**Freundschaftsspiel,*
***#Datum#**14.November 2012*
***#Details#**Linz, Linzer Stadion, 13.832 Zuschauer,*
***#Schiedsrichter#**Kralovec (CZE)*
***#Aufstellung#**(GK 24) Hein4 Lindner/FK Austria Wien – (2) György Garics/FC Bologna, (3) Aleksandar Dragovic/FC Basel, (4) Emanuel Pogatetz/VfL Wolfsburg, (13) Markus Suttner/FK Austria Wien – (7) Marko Arnautovic/SV Werder Bremen, (18) Christoph Leitgeb/FC RB Salzburg, (8) David Alaba/FC Bayern München, (16) Jakob Jantscher/Dynamo Moskau – (6) Andreas Ivanschitz/1.FSV Mainz 05 – (21) Marc Janko/Trabzonspor*
***#Austausch#**(17) Florian Klein/FC RB Salzburg für Garics (46.), (15) Sebastian Prödl/SV Werder Bremen für Pogatetz (46.), (14) Julian Baumgartlinger/1.FSV Mainz 05 für Leitgeb (46.), (19) Veli Kavlak/Besiktas JK für Alaba (59.), (9) Andreas Weimann/Aston Villa FC für Ivanschitz (64.), (11) Martin Harnik/VfB Stuttgart für Jantscher (76.)*
***#Teamchef#**Marcel Koller*
***#AufstellungG#**Boubacar Barry/KSC Lokeren, Igor Lolo/Kuban Krasnodar, Kolo Touré/Manchester City, Sol Bamba/Trabzonspor, Arthur Boka/VfB Stuttgart – Romaric/Real Saragossa, Arouna Kone/Wigan Athletic, Cheick Tioté/Newcastle United, Didier Ya Konan/Hannover 96, Max Gradel/AS Saint-Étienne – Wilfried Bony/Vitesse Arnheim*
***#Austausch#**Ismael Traore/Stade Brestois für Bamba (46.), Lacina Traoré/Anzhi Makhachkala für Bony (46.), Abdul Razak/Manchester City für Tiote (53.), Didier Drogba/Shanghai Shenhua für A. Kone (59.), Yaya Touré/Manchester City für Romaric, Salomon Kalou/Lille OSC für Ya Konan*
***#Teamchef#**Sabri Lamouchi*
***#Tore#**0:1 Ya Konan (44.) 0:2 Drogba (52.) 0:3 L. Traore (76.)*

Die gewählte Strategie

Jedes Feld in der späteren Tabelle **Spiel** ist eine Zeile in der Textdatei. Jede dieser Zeilen wird durch den späteren Feldnamen ***#Name#*** eingeleitet. Zwischen Spielen steht die Zeile *####*. Die Felder **Aufstellung**, **Tore** und **Austausch** sind vorläufig noch unstrukturierte Texte, die später, wenn aus diesem Text bereits die Tabelle **Spiel** geworden ist, per Programm in weitere Tabellen umgewandelt werden.

Wie formt man nun die Angaben des ÖFB zu den Spielen in die gezeigte Form mit den Rautezeichen um? Händisch wäre das ein mühsamer Weg. Man kann sich dafür aber hervorragende Hilfe in Form der ***Regular Expressions*** holen, die in Editoren für Programmier-Aufgaben und sogar in Word enthalten sind. Für diese Aufgabe verwende ich gerne den kostenlosen Editor ***Notepad++***. Im Dialogfeld von Notepad++ kann man einstellen, ob der Suchbegriff als ein ***Regulärer Ausdruck*** interpretiert werden soll.

Regular Expressions

Ein einfaches Suchen und Ersetzen erlaubt nur die Suche nach einem konkreten Text, also zum Beispiel nach *„Länderspiel“* oder *„Österreich“* aber nicht nach einem beliebigen Text, der im Zusammenhang mit *„Länderspiel“* vorkommt, zum Beispiel *„25. Länderspiel“* oder *„Österreich-Ungarn“*. Das *„25.“* und *„Ungarn“* ist bei jedem Spiel anders und daher versagt die einfache Suche.

Hier kommen die ***„Regular Expressions“*** ins Spiel. Grundsätzlich würde ein normaler Text in einer ***Regular Expression*** ebenso behandelt werden wie in einer normalen Suche.

Bestimmte Sonderzeichen haben aber eine besondere Bedeutung. Diese Zeichen sind:

.:()[]^+*?-\|

Daher kann man nach diesen Zeichen nicht direkt suchen sondern nur, indem man ihnen einem Backslash voranstellt. Um also zum Beispiel nach einem Plus-Zeichen zu suchen, verwendet man \+.

Einfache Umformungen

Einfach sind Umformungen wie *„Tore:…“* im Originaltext, denn es genügt nach *„Tore:“* zu suchen und durch *„#Tore#“* zu ersetzen. Dazu gehören auch Schreibfehler, wie das erwähnte *„Polser“* statt *„Polster“*.

Alles, was man jetzt korrigieren kann, kann man einheitlich in der ganzen Datei ausführen. Zum Beispiel die Entfernung mehrfacher Spaces, mehrfacher Zeilenumbrüche. Meist muss man diese Kleinigkeiten auch am Ende der Umformungen noch einmal ausführen, weil durch die Umformung wieder mehrfache Spaces oder Zeilenumbrüche entstehen.

Hier unterscheiden sich die Regular Expressions nicht von der normalen Suche.

Wie nun diese Spezialzeichen zu interpretieren sind, zeigen die folgenden Beispiele und auch die Sonderseite über Regular Expressions.

Der gesamte Prozess dieser Umwandlung ist hier nicht exakt darstellbar. Es gibt sehr viele Ausnahmen, hervorgerufen durch Tippfehler (manchmal steht irgendwo ein Abstand, dann fehlt er) oder durch verschiedenartigen Aufbau der Zeilen, wie man im Vergleich des ersten und des letzten Spiels sieht. Die Herstellung der strukturierten Datei ist daher eine Mischung aus systematischen Umformungen und manuellen Korrekturen.

Begegnung

Die Begegnung steht meist an erster Stelle eines Spiels und wird von keinem Text begleitet, daher muss man sich an dem *„Österreich“* orientieren. Aus *„Ungarn-Österreich“* oder *„Österreich-Ungarn“* soll daher werden *#Begegnung#Ungarn-Österreich* oder *#Begegnung#Österreich-Ungarn*, unabhängig davon, welchen Gegner Österreich hat. Solche Ersetzungen sind nun mit den Mitteln des einfachen *„Suchen und Ersetzen“* nicht mehr durchführbar und man benötigt die Hilfe der ***Regular Expressions***.

In dem Beispiel suchen wir nach einem Wort, das wir unverändert auch nach dem Ersetzungsvorgang verwenden wollen. Dafür verwendet man in der Regex-Sprache als Suchbegriff die Suche

(\w+)\-(Österreich)

und ersetzt durch

#Begegnung#\1\2\r\n#Gegner#\2\r\n#Heim#A\r\n

Diese „Formeln“ wollen erklärt werden:

(\w+)\-(Österreich)

() Eingeklammerte Begriffe werden in der Reihenfolge ihres Auftretens als Variable betrachtet, die als **\1**, **\2**… im Ersetzungsbegriff verwendet werden können.

\w Steht für irgendein alphanumerisches Zeichen, Pluszeichen **+** bedeutet, dass das Zeichen davon beliebig oft, mindestens aber einmal vorkommt. Das **\w+** ist eingeklammert, daher werden alle dadurch gefundenen Zeichen in die Variable **\1** gespeichert. Nach dem Bindestrich kann nicht direkt gesucht werden, daher muss ihm in der Suche ein Backslash vorangestellt werden.

####\r\n#Begegnung#\1\2\r\n#Gegner#\1\r\n#Heim#A\r\n

\1 steht für den Gegner und **\2** für „Österreich“. **####**, **#Begegnung#**, **#Gegner#** und **#Heim#A** werden unverändert ausgegeben. **\r** steht für das Steuerzeichen CR und **\n** für LF.

Durch diese Ersetzung entstehen die vier Felder **####**, **Begegnung**, **Gegner** und **Heim**. Der Zeilenrest nach den #...#-Feldbezeichnungen ist der eigentliche Feldinhalt und wird in die Datenbanktabelle **Spiel** importiert. Die Extra-Zeile **####** ist die Trennung zwischen zwei Länderspielen, an der sich das spätere Einleseprogramm orientieren kann.

Fassen wir also zusammen. Durch diese Ersetzung werden aus der Zeile

Ungarn-Österreich

die Zeilen

####
#Begegnung#Ungarn-Österreich
#Gegner#Ungarn
#Heim#A

D.h. etwa die Hälfte der Spiele wird auf diese Weise umgeformt.

Diese Suche muss noch einmal durchgeführt werden und zwar für den Fall von Heimspielen, weil dort die beiden Gegner ihren Platz tauschen. Die Such- und Ersetzungkommandos lauten dann:

Suchen
(Österreich)\-(\w+)
Ersetzen
####\r\n#Begegnung#\1\2\r\n#Gegner#\2\r\n#Heim#H\r\n

Grundsätzlich könnte man mit den Mitteln der Regular Expressions sogar beide Ersetzungen in einem einzigen Vorgang durchführen, doch lohnt sich in diesem Projekt dieser zusätzliche

Aufwand nicht, weil alle diese „Formeln“ nur einmal verwendet werden.

Ergebnis

Das Ergebnis eines Spiels wird meist in zwei Zeilen angegeben. Die erste Zeile ist das Endergebnis, die zweite ist der eingeklammerte Halbzeitstand.

Suchen
(\d+)\:(\d+)\r\n\((\d+)\:(\d+)\)\r\n
Ersetzen
\r\n#Ergebnis#\1:\2\r\n#ErgebnisH#\3:\4\r\n

Aus **2:3(1:1)** wird
#Ergebnis#2:3
#ErgebnisH#1:1

Art

Suchbegriff
(Freundschaftsspiel|Weltmeisterschaft|Europameisterschaft|WM-Qualifikation|EM-Qualifikation|EC der Nationen|Olympia)

Ersetzungsbegriff **\r\n#Art#\1\r\n**

Der senkrechte Strich ist die Trennung zwischen Textalternativen. Ganz egal, welcher der Texte gefunden wird, immer wird das Feld **Art** generiert.

Der vorangestellte Zeilenumbruch beim Ersetzungsbegriff garantiert, dass die Feldbezeichnung jedenfalls in einer neuen Zeile beginnt.

Länderspiel

Jedes Spiel enthält eine Zeile, die etwa so ausschaut:

633. Länderspiel, Innsbruck (Tivoli-NEU), 28.04.2004 Schiedsrichter: Damir Skomina, SLO

Aus diesen Zeilen werden die Felder **Spiel**, **Datum**, **Schiedsrichter** und **Stadion** extrahiert.

Leider ist der Aufbau des Dokuments nicht ganz gleich, daher muss man in Einzelschritten vorgehen und kann sich nicht darauf verlassen, diese Zeile in einem einzigen Schritt umwandeln zu können.

Suchen
(\d+)\. Länderspiel(.*)
Ersetzen
\r\n#Spiel#\1\r\n#Details#\2\r\n

Das Wort „Länderspiel“ dient und lediglich zum Auffinden der richtigen Zeilen. Zuerst trennen wir die Spielnummer **(\d+)** vom Rest der Zeile **(.*)**. **\d** steht für eine Ziffer **\d+** steht für mindestens eine Ziffer. Der Punkt **.** steht für irgendein Zeichen, der Stern ***** bedeutet, dass davor stehende Zeichen beliebig oft und auch gar nicht vorkommen kann. Damit erkennen wir auch Zeilen, die überhaupt nur mit dem Wort *Länderspiel* enden.

Datum

Suchen
(\d+)\.(\d+)\.(\d\d\d\d)
Ersetzen
\r\n#Datum#\1. \2. \3\r\n

Es gibt auch Datumsangaben mit ausgeschriebenen Monatsangaben und die müssen in einem weiteren, etwas komplizierteren Schritt verarbeitet werden.

Schiedsrichter

Suchen
Schiedsrichter\:(\w+)
Ersetzen
\r\n#Schiedsrichter#\1\r\n

Im Feld **Detail** bleiben nach diesen Umformungen Angaben zum Stadion stehen.

Abgeschlossen werden diese Umformungen durch Entfernen mehrfacher Zwischenräume:

Suchen
\s+
Ersetzen
" " (ohne Anführungszeichen)

Und mehrfacher Zeilenumbrüche, die sich während der Ersetzungen durch Voranstellen von **\r\n** angehäuft haben.

Suchen
(\r\n)+
Ersetzen
\r\n

Diese Ersetzungen sind eine grobe Skizze, wie man von dem Rohtext `alles-original.txt` zu einem strukturierten Text `alles.txt` mit Feldangaben kommen kann.

In der Praxis ist die Zahl der erforderlichen Umwandlungsschritte deutlich höher und zwischendurch durch manuelle Textkorrekturen begleitet, die dadurch erforderlich werden, dass die Originaldateien Tippfehler enthalten aber auch dadurch, dass die Umwandlungen versehentlich eine Ersetzung an einem Ort ausführen, an dem das gar nicht gewünscht ist.

Import in Datenbank

Es gibt jetzt also eine strukturierte Textdatei `alles.txt`. Jede Zeile enthält ein Datenfeld der Form **#Feldname#Inhalt**. Ein Datensatz ist vom nächsten durch eine Zeile **####** getrennt. Die Reihenfolge der Datenfelder ist unerheblich. Fehlen Datenfelder, bleibt das entsprechende Datenbankfeld eben leer.

Programmwahl

Man könnte den Import der Texte mit dem in Access eingebauten Visual Basic ausführen. Man kann aber auch die ungemein flexible ***PowerShell*** verwenden, die alles, was das DotNet-Framework anbietet, auch nutzen kann. Wenn man daher routinemäßig mit C# arbeitet, ist die Verwendung der PowerShell naheliegend.

Man kann die PowerShell entweder mit dem in Windows vorhandenen ***Power Shell ISE*** *(Integrated Software Environment)* oder auch mit dem kostenlosen Tool ***Power Gui Script Editor*** verwenden. Mit gefällt ***Power Gui*** besser aber für den Anfang ist das egal.

Eine kurze Darstellung der Skriptsprache PowerShell zeigt die folgende Seite, eine etwas längere Darstellung ist bei der Webversion dieses Beitrags verfügbar.

Der Import der Daten erfolgt in mehreren Schritten:

- Tabelle **Spiel** durch sequentielles Auslesen der Text-Datei `alles.txt` füllen. (`TextDateiLesen.ps1`)
  Das Feld Spiel (in der Textdatei alles.txt wurde es ID genannt) dient als eindeutige Datensatzkennzeichnung und Verknüpfungsfeld zu den Tabellen Spieler-Spiel und Tore.
- Tabelle **Spieler-Spiel** aus dem Feld **Aufstellung** der Tabelle **Spiel** erstellen (`SpielerTabelleErstellen.ps1`)
- Die Tabelle **Spieler-Spiel** enthält zunächst noch sowohl den **Verein** als auch den **Spieler**, beides in vielfacher Anzahl. Durch Gruppierung nach dem **Spieler** und nach dem **Verein** wird daraus die Tabellen **Spieler-Verein** und aus dieser dann die Tabellen **Spieler** und **Verein** hergestellt. Löscht man danach die nicht mehr benötigten Felder sind die genauen Schritte nicht mehr erkennbar.
- Tabelle **Tore** aus dem Feld **Tore** ermitteln (`ToreTabelleErstellen.ps1`)
- Tabelle **Spieler-Tausch** aus dem Feld **Austausch** erstellen und dann die Spielminuten der bereits eingetragenen Feldspieler korrigieren (`SpielerTauschTabelleErstellen.ps1`).
  Die neuen Eintauschspieler an die Tabelle **Spieler** anfügen und die Spielminuten eintragen.
  Dieser relativ komplizierte Vorgang ist notwendig, weil die ÖFB-Daten zwischen der Aufstellung (die 11 Mann, die das Spiel beginnen) und den Wechselspielern unterscheidet. Für die Zählung der Zahl der Einsätze ist diese Unterscheidung eher unpraktisch wie man sieht. Daher müssen eben die Austauschspieler zu der Spielertabelle hinzugefügt werden und dabei die Beginn– und Endeminute der neuen Spieler als auch jener, die ausgetauscht werden, bestimmt werden.

Datenquellen

http://www.oefb.at (Spiel 1..716)

http://www.weltfussball.at (Spiel 717-727)

Verwendete Programme

- Microsoft Access 2013
- Microsoft Excel 2013
- Notepad++
- Power GUI Script Editor
- Windows Power Shell ISE

Download

Webversion “Vermessung des Runden Leders”
http://d.pcnews.at/ins/pcn/139/000500/

- `Österreich.accdb` Datenbank
- `Österreich.xlsx` Grafische Auswertungen
- `Serien.docx` Textversion aller Serien

Webversion “Datamining”
http://d.pcnews.at/ins/pcn/139/002000/

- `Alles-Original.pdf`
- `Alles-Original.txt`
- `Alles.txt`
- `TextDateiLesen.ps1`
- `SpielerTabelleErstellen.ps1`
- `SpielerTauschTabelleErstellen.ps1`
- `ToreTabelleErstellen.ps1`

Links

Einführung in Access von Karel Štípek in PCNEWS-103

http://d.pcnews.at/ins/pcn/103/002700/main.htm

Und alle anderen Artikel von Karel Štípek

http://pcnews.at/?Show=AutorenArtikel&n=17484&id=13725&detail=%C5%A0

TextDateiLesen

Der Abschnitt `#Connection` stellt die Verbindung zur Access-Datenbank her.

Im Abschnitt `#Tabelle Games` werden die beiden benötigten Zugriffsmechanismen `SELECT` und `INSERT` für ein so OleDb-Adapter-Objekt `$Adapter` formuliert und für jedes Feld ein eigenes Parameter-Objekt definiert. Der Einfachheit halber werden alle Felder als Text importiert. Die Umwandlung in Zahlen (beim Feld `Spiel`) erfolgen später in der Datenbank durch Änderung des Feldtyps.

An dieser Stelle sieht man auch den nicht ganz reibungsfreien Entwicklungsvorgang. Anfangs wurde nämlich die Tabelle der Spiele `Games` genannt, damit sie mit einer bereits bestehenden Fußball-Datenbank zusammenpasst. Dann aber, als dieser Artikel entstehen sollte, wurde die Tabelle auf das deutsche `Spiele` umgeändert. Auch das sollte nicht bestehen bleiben und wurde nach Fertigstellung der Datenbank auf die Einzahl `Spiel` geändert. Aus Termingründen war es dann aber nicht mehr möglich, die Einzahlbildung auf die anderen Tabellen auszudehnen und daher ist zum Beispiele die Tabelle `Tore` immer noch in der Mehrzahl geschrieben.

Dieses Adapter-Objekt `$Adapter` kommuniziert mit einer `$DataTable`, die die Spaltennamen und Spaltentypen vom Adapter in `$Adapter.Fill ($DataTable)` übernimmt. Anfangs ist diese Tabelle leer, lediglich die Anzahl und die Namen der Spalten ist definiert.

Die Datei `alles.txt` wird in die Variable `$Alles` eingelesen, eine neue Datenzeile `$DataRow` angelegt.

Die for-Schleife `for($i=0; $i -lt $Alles.length; $i++)` liest alle Zeilen des Textes und überprüft den Aufbau der Zeilen.

Beginnt eine Zeile mit `####`, wir die Variable `$Game` hochgezählt, der mit Daten gefüllte Datensatz `$DataRow` an die Tabelle `$DataTable` angefügt und ein neuer, leerer Datensatz `$DataRow` erzeugt und alle Werte in diesem Datensatz mit der Funktion `ClearValues` gelöscht.

Beginnt der Datensatz mit `#Spiel#` wird zur Kontrolle ein Text ausgegeben.

Die Regular Expression `$regex = [regex] '#(.+)#(.*)'` findet Zeilen, die den gewählten Aufbau *#key#value* hat und isoliert daraus die Variablen `$key` und `$value`.

Bei den Keys `aufstellung`, `austausch` und `teamchef` muss noch der Umstand gelöst werden, dass es bei den aktuellen Begegnungen immer zwei gibt: die der Heimmannschaft und die der Auswärtsmannschaft. Bei der Gastmannschaft wird ein `G` zum Feldnamen angehängt.

Diese Schleife füllt daher 716 Datensätze in die Datentabelle. Die Übertragung in die Datenbank erfolgt mit dem Kommando `$Adapter.Update ($DataTable)`. Das genügt, denn wie die Tabelle in die Datenbank zu übertragen ist, wurde bereits in der Definition des Adapters genau festgelegt. Um diese konkreten Schreibvorgänge, die einfach eine schleifenartige Abarbeitung des `INSERT` -Kommandos ist, muss man sich nicht kümmern, das kann der Adapter ganz von allein.

```
#TextDateiLesen
#Connection
$DatabaseName = "s:\Desktop\nationalmannschaft\Österreich.accdb"
$ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=$DatabaseName;"
$Connection = New-Object System.Data.OleDb.OleDbConnection $ConnectionString
$Connection.Open()

#Tabelle Games
$SqlQuery = "SELECT * FROM Games"
$Command  = New-Object System.Data.OleDb.OleDbCommand $SqlQuery,$Connection
$SqlInsert = "INSERT INTO Games ( Spiel, Begegnung, Ergebnis, Pause, Art, Details, Datum,
Schiedsrichter, Aufstellung, AufstellungG, Austausch, AustauschG, Teamchef, TeamchefG, Tore ) VALUES
(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
$CommandInsert  = New-Object System.Data.OleDb.OleDbCommand $SqlInsert,$Connection
$Adapter = New-Object System.Data.OleDb.OleDbDataAdapter $Command
$Adapter.InsertCommand = $CommandInsert
$Parameter1 = New-Object System.Data.OleDb.OleDbParameter "@Spiel", Char, 255, "Spiel"
$Parameter2 = New-Object System.Data.OleDb.OleDbParameter "@Begegnung", Char, 255, "Begegnung"
$Parameter3 = New-Object System.Data.OleDb.OleDbParameter "@Ergebnis", Char, 255, "Ergebnis"
$Parameter4 = New-Object System.Data.OleDb.OleDbParameter "@Pause", Char, 255, "Pause"
$Parameter5 = New-Object System.Data.OleDb.OleDbParameter "@Art", Char, 255, "Art"
$Parameter6 = New-Object System.Data.OleDb.OleDbParameter "@Details", Char, 255, "Details"
$Parameter7 = New-Object System.Data.OleDb.OleDbParameter "@Datum", Char, 255, "Datum"
$Parameter8 = New-Object System.Data.OleDb.OleDbParameter "@Schiedsrichter", Char, 255,
"Schiedsrichter"
$Parameter9 = New-Object System.Data.OleDb.OleDbParameter "@Aufstellung", Char, 2550, "Aufstellung"
$Parameter10 = New-Object System.Data.OleDb.OleDbParameter "@AufstellungG", Char, 2550,
"AufstellungG"
$Parameter11 = New-Object System.Data.OleDb.OleDbParameter "@Austausch", Char, 255, "Austausch"
$Parameter12 = New-Object System.Data.OleDb.OleDbParameter "@AustauschG", Char, 255, "AustauschG"
$Parameter13 = New-Object System.Data.OleDb.OleDbParameter "@Teamchef", Char, 255, "Teamchef"
$Parameter14 = New-Object System.Data.OleDb.OleDbParameter "@TeamchefG", Char, 255, "TeamchefG"
$Parameter15 = New-Object System.Data.OleDb.OleDbParameter "@Tore", Char, 255, "Tore"
[void]$CommandInsert.Parameters.Add($Parameter1);
[void]$CommandInsert.Parameters.Add($Parameter2);
[void]$CommandInsert.Parameters.Add($Parameter3);
[void]$CommandInsert.Parameters.Add($Parameter4);
[void]$CommandInsert.Parameters.Add($Parameter5);
[void]$CommandInsert.Parameters.Add($Parameter6);
[void]$CommandInsert.Parameters.Add($Parameter7);
[void]$CommandInsert.Parameters.Add($Parameter8);
[void]$CommandInsert.Parameters.Add($Parameter9);
[void]$CommandInsert.Parameters.Add($Parameter10);
[void]$CommandInsert.Parameters.Add($Parameter11);
[void]$CommandInsert.Parameters.Add($Parameter12);
[void]$CommandInsert.Parameters.Add($Parameter13);
[void]$CommandInsert.Parameters.Add($Parameter14);
[void]$CommandInsert.Parameters.Add($Parameter15);
$DataTable = New-Object System.Data.DataTable
[void] $Adapter.Fill($DataTable)

$File_Alles = "S:\desktop\nationalmannschaft\alles.txt"
$Alles = Get-Content $File_Alles
$Game = 0
$KeyNames = @()
$DataRow = $DataTable.NewRow()
function ClearValues {
$DataRow["Spiel"]=""; $DataRow["Begegnung"]=""; $DataRow["Ergebnis"]=""; $DataRow["Pause"]="";
$DataRow["Art"]=""; $DataRow["Details"]=""; $DataRow["Datum"]=""; $DataRow["Schiedsrichter"]="";
$DataRow["Aufstellung"]=""; $DataRow["AufstellungG"]=""; $DataRow["Austausch"]=""; $DataRow
["AustauschG"]=""; $DataRow["Teamchef"]=""; $DataRow["TeamchefG"]=""; $DataRow["Tore"]=""
}
ClearValues

for($i=0; $i -lt $Alles.length; $i++) {
    $Line = $Alles[$i];
    #"Line:"+$Line
    if ($Line.StartsWith("####")) {
        $Game++

        $DataTable.Rows.Add($DataRow)

        $GameText = "Record:$Game "

        $DataRow = $DataTable.NewRow()
        ClearValues

        # if ($Game -eq 5) { break }
        continue
    }
    if ($Line.StartsWith("#Spiel#")) {
        Write-Host $GameText"Spiel"($Line.Substring(7))
        # if ($Line.Substring(7) -eq 1) { break }
    }
    $regex = [regex] '#(.+)#(.*)'
    $groups = $regex.Match($Line).Groups
    $key = $groups[1].ToString().ToLower()
    $value = $groups[2]

    $value = $value.ToString().Trim(' ',',',';',':')

    #Write-Host $key":"$value
    switch ($key) {
        "aufstellung" { if ($DataRow[$key] -eq "") { $DataRow[$key] = $value; }
             else { $DataRow[$key+"G" ] = $value; } }
        "austausch" { if ($DataRow[$key] -eq "") { $DataRow[$key] = $value; }
             else { $DataRow[$key+"G" ] = $value; } }
        "teamchef" { if ($DataRow[$key] -eq "") { $DataRow[$key] = $value; }
             else { $DataRow[$key+"G" ] = $value; } }
        default { $DataRow[$key] = $DataRow[$key] + $value; }
    }
}

$Adapter.Update($DataTable)
$Connection.Close()
```

SpielerTabelleErstellen

Der Abschnitt `#Datenbank`stellt die Verbindung zur Access-Datenbank her.

Im Abschnitt `#Tabelle Spiele` werden die beiden Felder benötigten Felder `Spiel` und `Aufstellung` in den Daten-Adapter `$AdapterSpiel` eingelesen.

In den zweiten Daten-Adapter `$AdapterSpieler` wird die Tabellenstruktur der Tabelle `Spieler` aufgebaut.

`$Team` aus der Tabelle `Spiel` enthält die gesamte Aufstellung als Text. Diese Textzeile wird in die einzelnen Spieler in der Array-Variablen `$TeamA` zerlegt.

Jeder dieser Einzelteile enthält ein bestimmtes Spielerformat. Bei neueren Aufstellungen wird der Verein nach dem Spielernamen durch einen Schrägstrich / angegeben. Die Rückennummer ist eingeklammert. Die zusätzliche Kennzeichnungen bei der Rückennummer: *G* steht für *Goalie* und *C* für *Captain*.

Bei der älteren Schreibweise ist der Verein eingeklammert und muss daher anders behandelt werden wie bei den neuere Spielen.

Spielernamen können einteilig sein (nur Familienname) oder zweiteilig (Vorname und Familienname). Leider ist diese Systematik nicht ganz durchgängig und manchmal stehen Vor– und Nachname in der falschen Reihenfolge. Da bleibt dann oft nichts anders übrig, als diese punktuellen Fehler händisch zu korrigieren.

Jedenfalls wird versucht, den Spielernamen in die Felder `FName` und `VName` richtig einzutragen und gleichzeitig auch die Rückennummer in das Feld `Nummer`.

In der Tabelle `Spieler` stehen die wesentlichen Felder: `Spiel`, `FName`, `Vname` und `Verein`. Jeder Datensatz ist ein Einsatz eines Spielers. Es gibt bis zum Spiel 727 9180 Einsätze. Diese jetzt importierte Tabelle Spiel wird später `Spieler-Spiel` heißen, weil daraus noch die Teiltabelle `Spieler-Verein` und dann daraus die Tabellen `Spieler` und `Verein` entstehen.

SpielerTauschTabelleErstellen

(Kode siehe nächste Seite) Leider ist der Spielertausch in einem anderen Format angegeben als die Aufstellung. Das Feld **Austausch** in der Tabelle **Spiele** wird daher zuerst in einer Hilfstabelle **Spieler_Spiel_Tausch** importiert. Da die Vorgänge dem Import der Spieler prinzipiell gleichen, wird dieser Vorgang nicht weiter beschrieben. Bei diesem Import werden die Bestandteile des Austausches (`SpielerEin`, `SpielerAus`, `VereinEin`, `VereinAus`, `Minute`) in einzelnen Felder abgelegt. Dann werden mit Mitteln von Access diese Datensätze dieser Tabelle **Spieler_Spiel_Tausch** mit der Anfüge-Abfrage **SpielerTausch Neue Spieler** an die Tabelle **Spieler-Spiel** angehängt und das Feld von-Minute auf den richtigen Zeitpunkt eingestellt. Die **bisMinute** ist 90. Danach wird bei jenen Spielern, die das Feld verlassen, die **bisMinute** auf denselben Wert eingestellt.

Anhang

Die beiden Programme **SpielerTauschTabelleErstellen** und **ToreTabelleErstellen** sind in einem Anhang der PDF-Version dieser Ausgabe auf den Seiten 33 und 34 enthalten.

```
#SpielerTabelleErstellen
#Datenbank
$DatabaseName = "s:\Desktop\nationalmannschaft\Österreich.accdb"
$ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=$DatabaseName;"
$Connection = New-Object System.Data.OleDb.OleDbConnection $ConnectionString
$Connection.Open()

#Tabelle Spiele
$SqlSpielQuery = "SELECT Spiel, Aufstellung FROM Spiele ORDER BY SPIEL"
$CommandSpiel  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielQuery,$Connection
$AdapterSpiel = New-Object System.Data.OleDb.OleDbDataAdapter $CommandSpiel

$DataTableSpiel = New-Object System.Data.DataTable
[void] $AdapterSpiel.Fill($DataTableSpiel)
$DataTableSpielLength = $DataTableSpiel.Rows.Count

#Tabelle Spieler
$SqlSpielerQuery = "SELECT * FROM Spieler"
$SqlSpielerInsert = "INSERT INTO Spieler (Spiel, FName, VName, Verein, Spielposition, Captain, Lfd, Nummer) VALUES
(@Spiel, @FName, @VName, @Verein, @Spielposition, @Captain, @Lfd, @Nummer )";
$CommandSpieler  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielerQuery,$Connection
$AdapterSpieler = New-Object System.Data.OleDb.OleDbDataAdapter $CommandSpieler
$AdapterSpieler.ContinueUpdateOnError = $false
$CommandSpielerInsert  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielerInsert,$Connection
$AdapterSpieler.InsertCommand = $CommandSpielerInsert
$Parameter1 = New-Object System.Data.OleDb.OleDbParameter "@Spiel", Integer, 4, "Spiel"
$Parameter2 = New-Object System.Data.OleDb.OleDbParameter "@FName", Char, 255, "FName"
$Parameter3 = New-Object System.Data.OleDb.OleDbParameter "@VName", Char, 255, "VName"
$Parameter4 = New-Object System.Data.OleDb.OleDbParameter "@Verein", Char, 255, "Verein"
$Parameter5 = New-Object System.Data.OleDb.OleDbParameter "@Spielposition", Char, 255, "Spielposition"
$Parameter6 = New-Object System.Data.OleDb.OleDbParameter "@Captain", Boolean, 4, "Captain"
$Parameter7 = New-Object System.Data.OleDb.OleDbParameter "@Lfd", Integer, 4, "Lfd"
$Parameter8 = New-Object System.Data.OleDb.OleDbParameter "@Nummer", Integer, 4, "Nummer"
[void]$CommandSpielerInsert.Parameters.Add($Parameter1)
[void]$CommandSpielerInsert.Parameters.Add($Parameter2)
[void]$CommandSpielerInsert.Parameters.Add($Parameter3)
[void]$CommandSpielerInsert.Parameters.Add($Parameter4)
[void]$CommandSpielerInsert.Parameters.Add($Parameter5)
[void]$CommandSpielerInsert.Parameters.Add($Parameter6)
[void]$CommandSpielerInsert.Parameters.Add($Parameter7)
[void]$CommandSpielerInsert.Parameters.Add($Parameter8)

$DTSpieler = New-Object System.Data.DataTable
[void] $AdapterSpieler.Fill($DTSpieler)

for ($i=0; $i -lt $DataTableSpielLength; $i++) {
    $Spiel = $DataTableSpiel.Rows[$i]["Spiel"].ToString()
    $Team = $DataTableSpiel.Rows[$i]["Aufstellung"].ToString()
    $Spiel+":"+$Team
    $Split_Spieler = ',',';'
    $Trim_Verein_Old = ' ',')'
    $TeamA = $Team.ToString().Split($Split_Spieler)
    #$TeamA
    for ($j=0; $j -lt $TeamA.Length; $j++) {
        $TeamA[$j] = $TeamA[$j].Trim()
    }
    for ($j=0; $j -lt $TeamA.Length; $j++)
    {
        $SpielerNummer = 0
        $Spieler = ""
        $Verein = ""
        $Spielposition = ""
        $Captain = $false

        #if ($TeamA.Length -eq 0) {} else { $Spieler }
        if ($Team.Contains("/")) { # neuere Aufstellung mit Rückennummern
            $Spieler = ($TeamA[$j].Split('/'))[0].Trim(' ')
            if ($TeamA[$j].Split('/').Length -eq 2) {
                $Verein = ($TeamA[$j].Split('/'))[1].Trim(' ')
            }
            if ($TeamA[$j].Contains("(")) { # Rückennummer ist angegeben
                $SpielerNummer = $Spieler.Substring(1,$Spieler.IndexOf(')')-1).Trim()
                if ($SpielerNummer.Contains("C")) { $SpielerNummer = $SpielerNummer.Replace("C",""); $Captain=$true }
                if ($SpielerNummer.Contains("GK"))
                    { $SpielerNummer = $SpielerNummer.Replace("GK",""); $Spielposition="G" }
                $Spieler = $Spieler.Substring($Spieler.IndexOf(')')+1).Trim()
            }
        } else { # alte Aufstellung
            if ($TeamA[$j].Contains("(")) { # Verein ist angegeben
                $Spieler = ($TeamA[$j].Split('(')[0].Trim(' ')
                $Verein = ($TeamA[$j].Split('('))[1].Trim($Trim_Verein_Old)
            } else { # Verein fehlt
                $Spieler = $TeamA[$j].Trim(' ')
            }
        }
        if ($j -eq 0) { $Spielposition="G" }
        $Out = $Spiel+":"+$Spieler+"->"+$Verein+"->"+$SpielerNummer
        if ($Captain -eq "true") {$Out +=" C"}
        if ($Spielposition -ne "") {$Out +=" "+$Spielposition}
        #$Out

        $DataRow = $DTSpieler.NewRow()
        $DataRow["Spiel"] = $Spiel
        $SpielerTeilname = $Spieler.Split(' ')
        switch ($SpielerTeilname.Count)
        {
            2 { if ($SpielerTeilname[1].Length -lt 3) {
                $DataRow["FName"] = $SpielerTeilname[0]
                $DataRow["VName"] = $SpielerTeilname[1]
                } else {
                $DataRow["FName"] = $SpielerTeilname[1]
                $DataRow["VName"] = $SpielerTeilname[0]
                }
            }
            default {
                $DataRow["FName"] = $Spieler
                $DataRow["VName"] = ""
            }
        }
        $DataRow["Verein"] = $Verein
        $DataRow["Spielposition"] = $Spielposition
        $DataRow["Captain"] = $Captain
        $DataRow["Lfd"] = $j + 1
        $DataRow["Nummer"] = [int]$SpielerNummer

        $DTSpieler.Rows.Add($DataRow)
    }
}
$AdapterSpieler.Update($DTSpieler)
$Connection.Close()
```

```
#SpielerTauschTabelleErstellen
#Datenbank
$DatabaseName = "s:\Desktop\nationalmannschaft\Österreich.accdb"
$ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=$DatabaseName;"
$Connection = New-Object System.Data.OleDb.OleDbConnection $ConnectionString
$Connection.Open()
#Tabelle Spiele
$SqlSpielQuery = "SELECT Spiel, Austausch FROM Spiele ORDER BY SPIEL"
$CommandSpiel  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielQuery,$Connection
$AdapterSpiel = New-Object System.Data.OleDb.OleDbDataAdapter $CommandSpiel
$DataTableSpiel = New-Object System.Data.DataTable
[void] $AdapterSpiel.Fill($DataTableSpiel)
$DataTableSpielLength = $DataTableSpiel.Rows.Count
#Tabelle Spieler-Spiel-Tausch
$SqlSpielerQuery = "SELECT * FROM Spieler_Spiel_Tausch"
$SqlSpielerInsert = "INSERT INTO Spieler_Spiel_Tausch
            (Spiel, Tausch, Lfd, TauschMinute, SpielerEinF, SpielerEinV, SpielerEin_Verein, SpielerEin_Nummer, SpielerAusF, SpielerAusV)
    VALUES (@Spiel, @Tausch, @Lfd, @TauschMinute, @SpielerEinF, @SpielerEinV, @SpieTerEin_Verein, @SpielerEin_Nummer, @SpielerAusF, @SpielerAusV )";
$CommandSpieler  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielerQuery,$Connection
$AdapterSpieler = New-Object System.Data.OleDb.OleDbDataAdapter $CommandSpieler
$AdapterSpieler.ContinueUpdateOnError = $false
$CommandSpielerInsert  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielerInsert,$Connection
$AdapterSpieler.InsertCommand = $CommandSpielerInsert
$Parameter1 = New-Object System.Data.OleDb.OleDbParameter "@Spiel", Integer, 4, "Spiel"
$Parameter2 = New-Object System.Data.OleDb.OleDbParameter "@Tausch", Char, 2500, "Tausch"
$Parameter3 = New-Object System.Data.OleDb.OleDbParameter "@Lfd", Integer, 4, "Lfd"
$Parameter4 = New-Object System.Data.OleDb.OleDbParameter "@TauschMinute", Integer, 4, "TauschMinute"
$Parameter5 = New-Object System.Data.OleDb.OleDbParameter "@SpielerEinF", Char, 255, "SpielerEinF"
$Parameter6 = New-Object System.Data.OleDb.OleDbParameter "@SpielerEinV", Char, 255, "SpielerEinV"
$Parameter7 = New-Object System.Data.OleDb.OleDbParameter "@SpielerEin_Verein", Char, 255, "SpielerEin_Verein"
$Parameter8 = New-Object System.Data.OleDb.OleDbParameter "@SpielerEin_Nummer", Integer, 4, "SpielerEin_Nummer"
$Parameter9 = New-Object System.Data.OleDb.OleDbParameter "@SpielerAusF", Char, 255, "SpielerAusF"
$Parameter10 = New-Object System.Data.OleDb.OleDbParameter "@SpielerAusV", Char, 255, "SpielerAusV"
[void]$CommandSpielerInsert.Parameters.Add($Parameter1)
…// wiederholen bis Parameter 10

$DTSpieler = New-Object System.Data.DataTable
[void] $AdapterSpieler.Fill($DTSpieler)

function GetFName {
    if ($args[0].Contains(" ")) {
        $Namensteile = $args[0].ToString().Split(' ')
        if ($Namensteile[1].Length -le 2) { # Vorname nachgestellt
            $Namensteile[0]
        } else {
            $Namensteile[1]
        }
    } else {
        $args[0]
    }
}
function GetVName {
    if ($args[0].Contains(" ")) {
        $Namensteile = $args[0].ToString().Split(' ')
        if ($Namensteile[1].Length -le 2) { # Vorname nachgestellt
            $Namensteile[1]
        } else {
            $Namensteile[0]
        }
    } else {
        ""
    }
}
for ($i=0; $i -lt $DataTableSpielLength; $i++) {
    $Spiel = $DataTableSpiel.Rows[$i]["Spiel"].ToString()
    $TauschAlle = $DataTableSpiel.Rows[$i]["Austausch"]
    if ($TauschAlle -eq $null) { $TauschAlle = "" }
    if ($TauschAlle -ne "") {

        "#"+$Spiel
        $TauschAlleArray = $TauschAlle.ToString().Split(',')
        for ($j=0; $j -lt $TauschAlleArray.Length; $j++)
        {
            $Tausch = $TauschAlleArray[$j]

            $Minute = 0
            $regexMinute = [regex] '\((\d+)\.\)'
            $groups = $regexMinute.Match($Tausch).Groups
            if ($groups.Count-eq 2) {
                $Minute = [int]$groups[1].Value
                $Tausch = $regexMinute.Replace($Tausch,"").Trim()
            } else { $Minute=999 }
            $SpielerEin = ""
            $VereinEin = ""
            $SpielerAus = ""
            $SpielerNummer = 0
            $regexNameVerein = "\((.*)\)" #verein kann auch leer sein
            $regexNameSpieler = "(.+)"
            if ($TauschAlle.Contains("/")) { # neuere Aufstellung mit Rückennummern
                if ($Tausch.Contains("/")) { #Verein angegeben
                    $regexTausch = [regex]($regexNameSpieler+" *?\/ *?"+$regexNameSpieler+" *?(statt|für) +?"+$regexNameSpieler)
                    $groups = $regexTausch.Match($Tausch).Groups
                    if ($groups.Count-eq 5) {
                        $SpielerEin = $groups[1].Value
                        $VereinEin = $groups[2].Value
                        $SpielerAus = $groups[4].Value
                    } else {
                        $SpielerEin="nicht gefunden"
                    }
                } else {
                    $regexTausch = [regex]($regexNameSpieler+" +?(statt|für) +?"+$regexNameSpieler)
                    $groups = $regexTausch.Match($Tausch).Groups
                    if ($groups.Count-eq 4) {
                        $SpielerEin = $groups[1].Value
                        $SpielerAus = $groups[3].Value
                    } else {
                        $SpielerEin="nicht gefunden"
                    }
                }
                if ($SpielerEin.Contains("(")) { # Rückennummer ist angegeben
                    $SpielerNummer = $SpielerEin.Substring(1,$SpielerEin.IndexOf(')')-1).Trim()
                    $SpielerEin = $SpielerEin.Substring($SpielerEin.IndexOf(')')+1).Trim()
                }
            } else { # alte Aufstellung
                if ($Tausch.Contains("(")) { #Verein angegeben
                    $regexTausch = [regex]($regexNameSpieler+" +?"+$regexNameVerein+" *?für +?"+$regexNameSpieler)
                    $groups = $regexTausch.Match($Tausch).Groups
                    if ($groups.Count-eq 4) {
                        $SpielerEin = $groups[1].Value
                        $VereinEin = $groups[2].Value
                        $SpielerAus = $groups[3].Value
                    } else {
                        $SpielerEin="nicht gefunden"
                    }
                } else {
                    $regexTausch = [regex]($regexNameSpieler+" +?für +?"+$regexNameSpieler)
                    $groups = $regexTausch.Match($Tausch).Groups
                    if ($groups.Count-eq 3) {
                        $SpielerEin = $groups[1].Value
                        $SpielerAus = $groups[2].Value
                    } else {
                        $SpielerEin="nicht gefunden"
                    }
                }
            }
            $SpielerEin = $SpielerEin.Trim()
            $SpielerEinF = GetFName($SpielerEin)
            $SpielerEinV = GetVName($SpielerEin)

            $SpielerAus = $SpielerAus.Trim()
            $SpielerAusF = GetFName($SpielerAus)
            $SpielerAusV = GetVName($SpielerAus)

            $VereinEin = $VereinEin.Trim()

            $DataRow = $DTSpieler.NewRow()
            $DataRow["Spiel"] = $Spiel
            $DataRow["Tausch"] = $Tausch
            $DataRow["Lfd"] = [int]($j + 1)
            $DataRow["TauschMinute"] = [int]($Minute)
            $DataRow["SpielerEinF"] = $SpielerEinF
            $DataRow["SpielerEinV"] = $SpielerEinV
            $DataRow["SpielerEin_Nummer"] = $SpielerNummer
            $DataRow["SpielerEin_Verein"] = $VereinEin
            $DataRow["SpielerAusF"] = $SpielerAusF
            $DataRow["SpielerAusV"] = $SpielerAusV

            $DataRow["ID_SpielerAus"] = -1
            $DataRow["ID_SpielerEin"] = -1
            $DataRow["ID_Verein"] = -1

            $DTSpieler.Rows.Add($DataRow)

            [string]
$Minute+":"+$SpielerEinV+$SpielerEinF+"("+$SpielerNummer+")"+"/"+$VereinEin+"<-
>"+$SpielerAusV+$SpielerAusF
        }
    }
}
$AdapterSpieler.Update($DTSpieler)
$Connection.Close()
```

ToreTabelleErstellen

Der Abschnitt `#Datenbank` stellt die Verbindung zur Access-Datenbank her, die Tabelle `Spiel` wurde bereits angelegt und daraus werden die Felder `Spiel` und `Tore` selektiert und über den Adapter `$AdapterSpiel` dem Programm zur Verfügung gestellt. Das Feld `Spiel` muss auch in der neuen Tabelle `Tore` enthalten sein, damit zwischen den Tabellen `Spiele` und `Tore` eine Relation hergestellt werden kann.

Die Tabelle `Tore` wird durch den Adapter `$AdapterSpieler` abgebildet (ja, ja, das kommt vom Kopieren, denn hier sollte natürlich `$AdapterTore` stehen aber durch die Übernahme vom vorigen Import-Programm wurde dieser sinnstörende Fehler in der Bezeichnung der Variablen übersehen). Man sieht, dass die einzelnen Felder verschiedene Typen aufweisen.

Die Auswertung des Feldes Tore ist leider nicht „geradlinig“, denn man muss zwischen Formaten unterscheiden, die bei den frühen Spielen verwendet worden sind.

Zuerst wird die Zeile an den Beistrichen zerteilt `$ToreA = $Tore.Split(',')`. Wenn der Spielstand einen Doppelpunkt enthält, war der Spielstand angegeben `$Tore.Contains(":")` (das sind die jüngeren Spiele).

Abgesehen von den Zeichenklaubereien, die zeigen, wie man mit bestimmten wechselnden Umständen umgehen muss, ist wichtig dass die Zeilenauswertung entweder erfolgreich ist oder nicht. Im Erfolgsfall wird in der Hilfsfunktion `TorSpeichern` die erfolgreiche Speicherung des Tors ausgegeben. Im Fehlerfall wird aber eine bunte Diagnosezeile ausgegeben.

An dieser Stelle sieht man, dass dieses Umwandeln des Textes im `Tore`-Feld mit sehr vielen Schreibfehlern im Originaltext zu kämpfen hat, die man alle erst durch Editieren beseitigen muss, bis schließlich alle `Tore`-Zeilen fehlerfrei gelesen werden.

Aus diesen wiederholten Einleseversuchen resultiert auch, dass viele Primärschlüsselfelder nicht beim Wert 1 beginnen, wie man eigentlich vermuten würde sondern bei sehr hohen Werten, je nachdem, wie oft die eingelesenen Daten wieder gelöscht worden sind. Die Primärschlüsselfelder beginnen nämlich nur beim ersten Versuch beim Wert 1. Wird ein Datensatz gelöscht (oder eben mehrere), dann werden diese Werte nicht wieder vergeben und die Zählung beginnt bei höheren ID-Werten.

Mit dem Einlesen der Tore ist es aber leider nicht getan. Jetzt wartet noch Arbeit in der Datenbank, denn die Verfasser der Länderspiel-Berichte haben die Spieler-Namen in der Aufstellung und in der Torschützenliste nicht gleich geschrieben. Stand also in der Aufstellung noch *Anton Polster*, hat man das in der Torschützenliste auf *Polster* verkürzt. Mit der Konsequenz, dass das Programm diesen Umstand in der Form auswertet, dass es sich um zwei verschiedene Spieler handelt.

Bei einem konkreten Tor wird der Schütze durch den Spielernamen charakterisiert. Diesen Namen muss man in Aktualisierungs-Abfragen durch der richtigen Wert der **`ID_Spieler`** ersetzen. Wie man am Beispiel *Polster* gesehen hat, ist das oft kein eindeutiger Vorgang. Glücklicherweise wurden aber die Spieler der früheren Jahre nur mit dem Familiennamen eingetragen, sodass diese Fehler erst in den Spiele der letzten Jahre auftreten.

```
#Datenbank
$DatabaseName = "s:\Desktop\nationalmannschaft\Österreich.accdb"
$ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=$DatabaseName;"
$Connection = New-Object System.Data.OleDb.OleDbConnection $ConnectionString
$Connection.Open()

#Tabelle Spiel
$SqlSpielQuery = "SELECT Spiel, Tore FROM Spiele WHERE (((Tore)<>'')) ORDER BY SPIEL"
$CommandSpiel  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielQuery,$Connection
$AdapterSpiel = New-Object System.Data.OleDb.OleDbDataAdapter $CommandSpiel

$DataTableSpiel = New-Object System.Data.DataTable
[void] $AdapterSpiel.Fill($DataTableSpiel)
$DataTableSpielLength = $DataTableSpiel.Rows.Count

#Tabelle Tore
$SqlSpielerQuery = "SELECT * FROM Tore"
$SqlSpielerInsert = "INSERT INTO Tore
            (Spiel, TorMinute, ID_Spieler, Spieler, Spielstand, Penalty, Freekick)
    VALUES (@Spiel, @TorMinute, @ID_Spieler, @Spieler, @Spielstand, @Penalty, @Freekick )";
$CommandSpieler  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielerQuery,$Connection
$AdapterSpieler = New-Object System.Data.OleDb.OleDbDataAdapter $CommandSpieler
$AdapterSpieler.ContinueUpdateOnError = $false
$CommandSpielerInsert  = New-Object System.Data.OleDb.OleDbCommand $SqlSpielerInsert,$Connection
$AdapterSpieler.InsertCommand = $CommandSpielerInsert
$Parameter1 = New-Object System.Data.OleDb.OleDbParameter "@Spiel", Integer, 4, "Spiel"
$Parameter2 = New-Object System.Data.OleDb.OleDbParameter "@TorMinute", Integer, 4, "TorMinute"
$Parameter3 = New-Object System.Data.OleDb.OleDbParameter "@ID_Spieler", Integer, 4, "ID_Spieler"
$Parameter4 = New-Object System.Data.OleDb.OleDbParameter "@Spieler", Char, 255, "Spieler"
$Parameter5 = New-Object System.Data.OleDb.OleDbParameter "@Spielstand", Char, 255, "Spielstand"
$Parameter6 = New-Object System.Data.OleDb.OleDbParameter "@Penalty", Boolean, 4, "Penalty"
$Parameter7 = New-Object System.Data.OleDb.OleDbParameter "@Freekick", Boolean, 4, "Freekick"
[void]$CommandSpielerInsert.Parameters.Add($Parameter1)
[void]$CommandSpielerInsert.Parameters.Add($Parameter2)
[void]$CommandSpielerInsert.Parameters.Add($Parameter3)
[void]$CommandSpielerInsert.Parameters.Add($Parameter4)
[void]$CommandSpielerInsert.Parameters.Add($Parameter5)
[void]$CommandSpielerInsert.Parameters.Add($Parameter6)
[void]$CommandSpielerInsert.Parameters.Add($Parameter7)

$DTSpieler = New-Object System.Data.DataTable
[void] $AdapterSpieler.Fill($DTSpieler)

function TorSpeichern ($a, $b, $c, $d, $e, $f) {
    $DataRow = $DTSpieler.NewRow()

    $DataRow["Spiel"] = $a
    $DataRow["Spieler"] = $b.Trim()
    $DataRow["TorMinute"] = $c
    $DataRow["Spielstand"] = $d
    $DataRow["Penalty"] = $e
    $DataRow["Freekick"] = $f
    $DataRow["ID_Spieler"] = 0

    $DTSpieler.Rows.Add($DataRow)

    Write-Host "    "$c"-"$b"("$d")"
}
for ($i=0; $i -lt $DataTableSpielLength; $i++) {
    $Spiel = [int]$DataTableSpiel.Rows[$i]["Spiel"].ToString()
    $Tore = $DataTableSpiel.Rows[$i]["Tore"]

    Write-Host "#"$Spiel":"$Tore

    $ToreA = $Tore.Split(',')
    for ($j=0; $j -lt $ToreA.Length; $j++) {
        $TorSchuetze = $ToreA[$j].Trim()
        if ($Tore.Contains(":")) { #Spielstand wurde mitangegeben
            $Tore = $Tore.Replace(")",",").Trim(",")
            $Tore = $Tore.Replace(",,",",")
            $ToreA = $Tore.Split(',')
            $regexTorSchuetze = [regex] "(\d\:\d) (.*?) \((.+?)\.\)"
            $groups = $regexTorSchuetze.Match($TorSchuetze).Groups
            if ($match.Success) {
                $Minute = $groups[3].Value
                $Elfer = $Minute.Contains("E")
                $Freeekick= $Minute.Contains("F")
                $Minute = $Minute.Replace("E","")
                $Minute = $Minute.Replace("F","")
                TorSpeichern $Spiel $groups[2].Value $Minute  $groups[1].Value $Elfer $Freeekick
            } else {
                Write-Host "    ERROR" -ForegroundColor Red -BackgroundColor White
            }
        } else { # ohne Spielstand, nur die Minute und der Name
            if ($TorSchuetze.Contains("(")) { #enthält Minute
                $regexTorSchuetze = [regex] "(.*?) \((.+)\.\)"
                $groups = $regexTorSchuetze.Match($TorSchuetze).Groups
                if ($match.Success) {
                    $Torminuten = $groups[2].Value
                    $TorminutenA = $groups[2].Value.Split(' ')
                    for ($k=0; $k -lt $TorminutenA.Count; $k++) {
                        TorSpeichern  $Spiel  $groups[1].Value  $TorminutenA[$k].Replace('.','')    ""
$false $false
                    }
                } else {
                    Write-Host "    ERROR" -ForegroundColor Red -BackgroundColor White
                }
            } else { # nur Torschützen
                TorSpeichern $Spiel $ToreA[$j] 999 "" $false $false
            }
        }
    }
    Write-Host
}
$AdapterSpieler.Update($DTSpieler)
$Connection.Close()
```