```
#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()
```