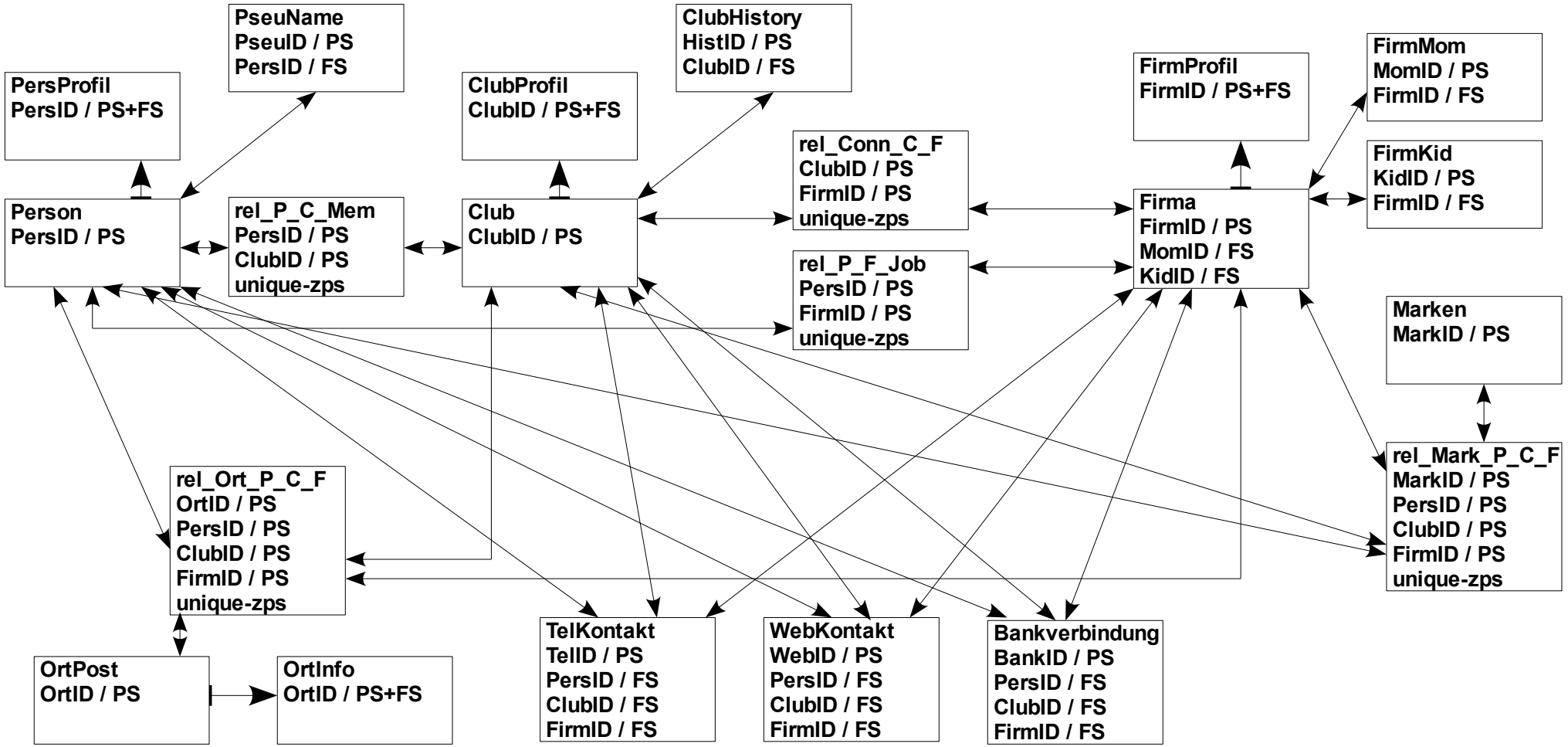
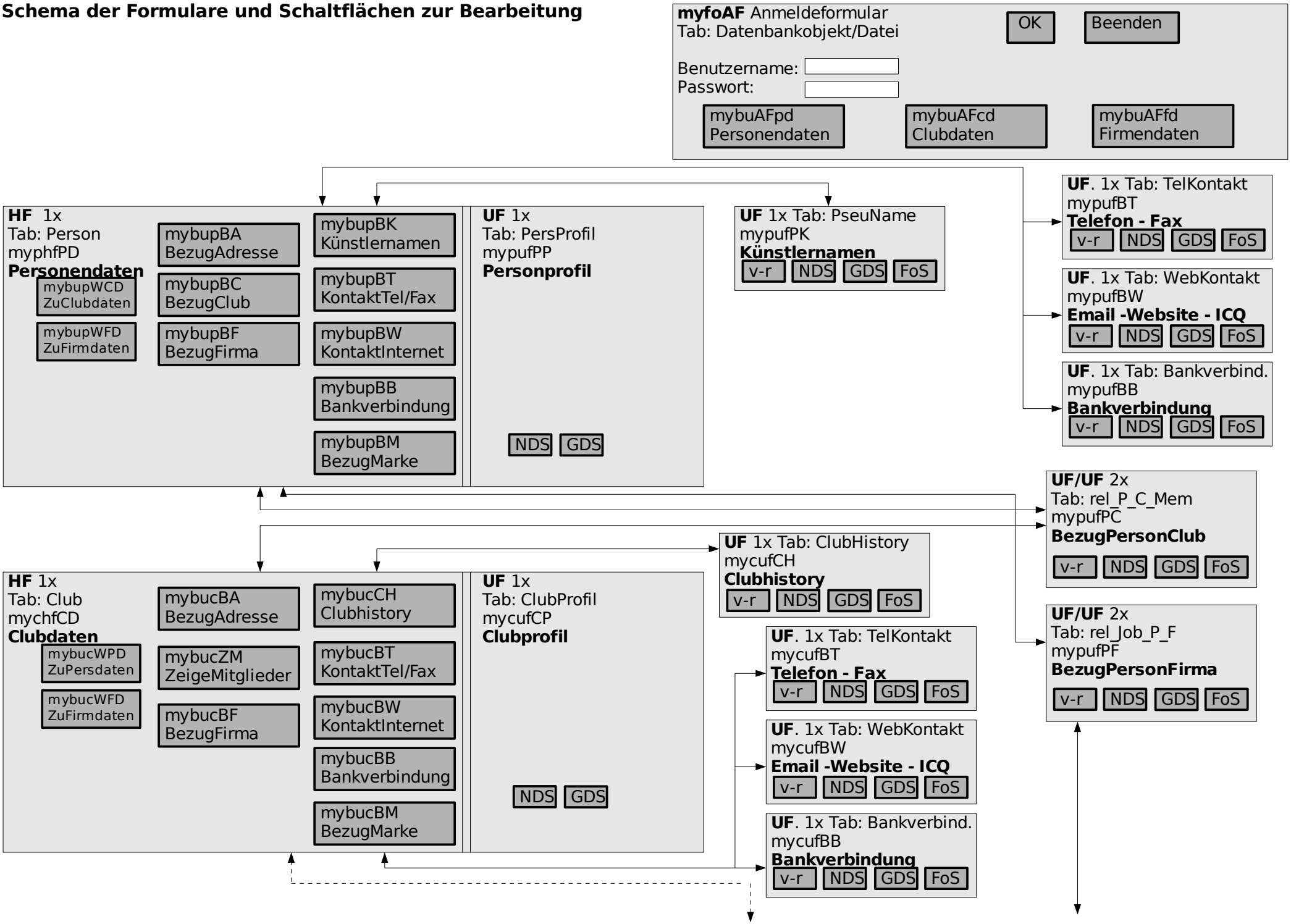
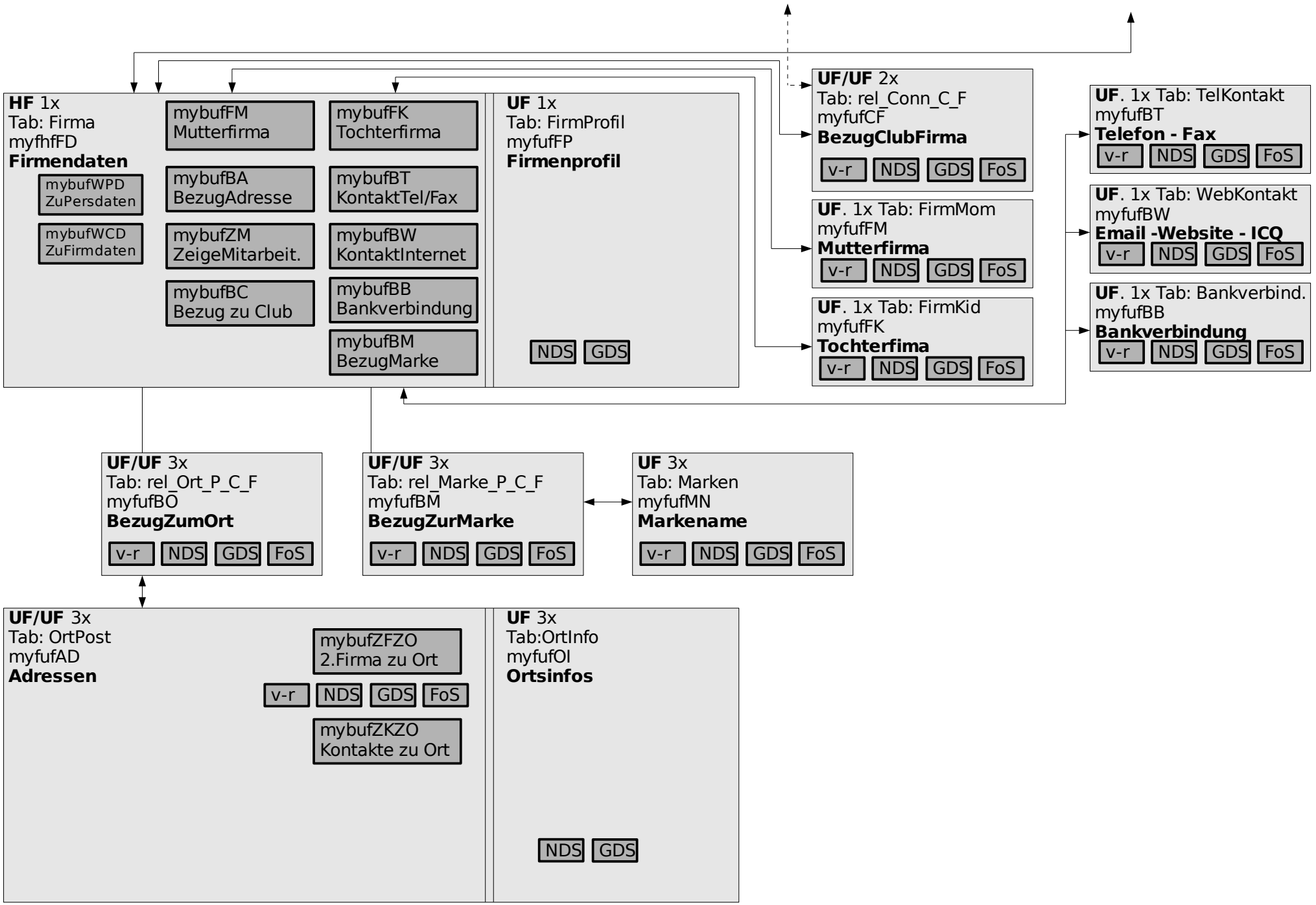
Tabellen- und Formularschema Adressen- Kontaktdatenbank

Schema der Tabellen mit Darstellung der Tabellenverknüpfungen (Relationen):



Schema der Formulare und Schaltflächen zur Bearbeitung




HF 1x
Tab: Firma
myfhfFD
Firmendaten
mybufWPD
ZuPersdaten
mybufWCD
ZuFirmdaten
mybufFM
Mutterfirma
mybufFK
Tochterfirma
mybufBA
BezugAdresse
mybufBT
KontaktTel/Fax
mybufZM
ZeigeMitarbeit.
mybufBW
KontaktInternet
mybufBC
Bezug zu Club
mybufBB
Bankverbindung
mybufBM
BezugMarke
UF 1x
Tab: FirmProfil
myfufFP
Firmenprofil
NDS GDS
UF/UF 2x
Tab: rel_Conn_C_F
myfufCF
BezugClubFirma
v-r NDS GDS FoS
UF. 1x Tab: FirmMom
myfufFM
Mutterfirma
v-r NDS GDS FoS
UF. 1x Tab: FirmKid
myfufFK
Tochterfima
v-r NDS GDS FoS
UF. 1x Tab: TelKontakt
myfufBT
Telefon - Fax
v-r NDS GDS FoS
UF. 1x Tab: WebKontakt
myfufBW
Email -Website - ICQ
v-r NDS GDS FoS
UF. 1x Tab: Bankverbind.
myfufBB
Bankverbindung
v-r NDS GDS FoS
UF/UF 3x
Tab: rel_Ort_P_C_F
myfufBO
BezugZumOrt
v-r NDS GDS FoS
UF/UF 3x
Tab: rel_Marke_P_C_F
myfufBM
BezugZurMarke
v-r NDS GDS FoS
UF 3x
Tab: Marken
myfufMN
Markename
v-r NDS GDS FoS
UF/UF 3x
Tab: OrtPost
myfufAD
Adressen
mybufZFZO
2.Firma zu Ort
v-r NDS GDS FoS
mybufZKZO
Kontakte zu Ort
UF 3x
Tab:OrtInfo
myfufOI
Ortsinfos
NDS GDS

Dieses Schema stellt den von mir angedachten Datenbankentwurf dar. Ich habe es erstellt um mir einerseits selbst eine Art „Roten Faden“ für die Umsetzung zu schaffen, andererseits um mir Rat und Hilfe hierfür zu suchen. Denn meine bisherigen Kenntnisse zu der ganzen Materie stammen nur aus 2 guten Büchern zum Thema (MySQL4.0 von Michael Kofler und OpenOffice2.0 von Günter Born, ein drittes zur Makroprogrammierung von Thomas Krumbein habe ich gerade angefangen zu lesen).

Danke an die Autoren, die Bücher sind wirklich informativ und gut geschrieben. Aber für einen absoluten Programmieranfänger wie mich ist doch noch nicht alles so richtig verständlich und auch die bisherige Recherche in entsprechenden Internetforen hat doch einige Fragen offen gelassen. Kurz noch ein Wort zum Autor dieses Beitrags, also meiner Wenigkeit:
Nickname henzmen, Ende 40, von Beruf Kaufmann und, durch den beruflichen Umgang damit, von MS-Access doch etwas „verwöhnt“. Wobei sich meine Access Kenntnisse aber eher auf das zusammen schieben von Icons beschränken, Erfahrung mir Programmieren in Visual-Basic oder ähnlichem ist gleich Null. Der Wunsch eine „richtige Datenbank“ zu erstellen beruht einfach auf der Notwendigkeit, Ordnung in einen enormen Adressenbestand zu bringen der sich, im Laufe vieler Jahre als Vereinsschriftführer, angesammelt hat und gegenwärtig noch in diversen Tabellen und Karteikarten verwaltet wird.

Meine folgenden Ausführungen beruhen daher auch nur auf dem, was ich bisher verstanden habe bzw. besser gesagt; Glaube verstanden zu haben.

Die Tabellenstruktur habe ich im Entwurfsmodus mit OOo-Base 2.3.1 erstellt, die Tabellen (Tabellentreiber ausschließlich My-ISAM) habe ich nachträglich (weil einfacher und sicherer) mit PHP-myadmin indiziert. Die eigentliche Datenbank läuft unter MySQL 5.03, auf Debian-Etch, die Verbindung zu OOo-Base besteht per ODBC Version 3.51. Die Anbindung funktioniert einwandfrei, die per OOo-Base erstellten Tabellen sind, so wie dort definiert, in php-myadmin sichtbar und auch zu administrieren. Mit ein Paar manuell eingetippten Testdaten habe ich die Tabellenverknüpfungen, mit einigen SELECT Befehlen in php-myadmin, getestet. Das funktioniert alles wie gedacht und gewünscht. Es geht jetzt quasi nur darum, eine komfortablere Formularstruktur als Frontend zu realisieren.

Bevor ich auf diese näher eingehe noch einige Anmerkungen zur Basis, also dem DB-Backend MySQL:

Die erste Grafik (nicht farblich abgehobene Felder) soll eine Überblick über die Tabellen und deren Relationen vermitteln. Jedes Feld steht für eine Tabelle, der Tabellenname steht jeweils in der ersten Zeile des Feldes, darunter folgt der Name des Primärschlüsselfeldes (mit **/ PS** als solches gekennzeichnet), wieder darunter sind das bzw. die möglichen Fremdschlüsselfelder der Tabelle (mit **/ FS** gekennzeichnet) dargestellt. Die Kennzeichnung **PS+FS** steht für Schlüsselfelder einer möglichen 1-1 bzw. sonst 1-0 Beziehung. Der Zusatz **unique-zps** kennzeichnet einen zusammengesetzten Primärschlüssel in den Relationstabellen (die alle mit „**rel**“ beginnen und eine n-m-Beziehung auf zwei 1-n-Beziehungen reduzieren) und die, neben den Schlüsselfeldern, nur einige Zusatzinformationen aufnehmen.

Ich habe dabei bewusst darauf verzichtet, die einzelnen Datenfelder jeder Tabelle anzugeben. Das Ganze wäre zu unübersichtlich für diesen Zweck, darum gebe ich nur die Tabellennamen und die Namen der Schlüsselfelder an. Beim Inhalt der Tabellen ist aber besonderer Wert darauf gelegt, alle Informationen möglichst Renundanz frei speichern zu können, hierzu ein Beispiel:

Eine Adresse (Postleitzahl, Ort, Straße mit Hausnummer, Haustelefonanschluss) kann sowohl zu einer Person (Single) als auch mehreren Personen (Familie) gehören. Gleichzeitig kann eine Person mehrere Adressen (z. B. zweiten Wohnsitz am Arbeitsort usw.) haben. Ebenso kann ein und die selbe Adresse zu einer Privatperson und gleichzeitig einem Verein (Vereinsbüro im Privathaus) gehören. Solche und ähnliche Konstellationen (wie z. B. auch die Abbildung der Beziehung einer Person zu einem Verein im Sinne von „seit/bis wann Mitglied, eventuell in welchem Vorstandsamt“, sind der Grund für dieses Datenmodell. Um das kurz deutlich zu machen:

„Max Mustermann“ kommt nur EIN mal in der Tabelle **Person** vor, egal ob er Mitglied in einem oder 5 Vereinen ist. Der Verein „Die Meistersinger“ kommt auch nur EIN mal in der Tabelle **Club** vor, egal ob er 10 oder 100 Mitglieder hat. Wer nun von bzw. bis wann und ggf. in welcher Funktion, Mitglied in welchem Verein ist, steht in der Tabelle **rel_P_C_Mem**.

Da es durchaus vorkommt, das zu einer Person bzw. einem Verein bestimmte Daten nicht bekannt, oder auch einfach nicht vorhanden sind (nicht jeder hat mehrere Telefonnummern, Emailadressen oder eine eigene Website), sind für solche Daten auch die dargestellten Zusatztabellen vorgesehen.

Diese Zusatztabellen haben lediglich den Zweck in den anderen, für die am häufigsten bekannten Daten vorgesehenen, Tabellen zu viele potentielle Leerfelder oder aber Wiederholungen im Sinne von „erste Handynummer, zweite Handynummer, Firmenhandynummer usw.“ zu vermeiden. Hierbei hatte ich zunächst auch an die Verwendung von Relationstabellen gedacht, hab dies aber verworfen und mich für die dargestellte Version entschlossen. Die gestattet es immerhin, z. B. die gleiche Telefonnummer einem Verein und einer Privatperson (ggf. sogar noch einer Firma) zu zuordnen.

Das **unique-zps** ist nicht zu verwechseln mit dem in MySQL auch gebräuchlichen **Unique-Index**. Ein solcher wird in einigen Tabellen (wie z.B. der Tabelle **OrtPost** über die Felder Postleitzahl, Ort, Straße mit Hausnummer, Haustelefonanschluss) auch verwendet, hier aber nur über Felder mit Werten (Nutzdaten) also nicht über Schlüsselfelder, und nur um „Doppelgänger“ zu vermeiden. Alle Schlüsselfelder sind als Integerzahlen definiert, die Primärschlüsselfelder der folgenden Tabellen sind hierbei mit dem Attribut auto-increment versehen:

Person, PseuName, Club, ClubHistory, Firma, FirmMom, FirmKid, OrtPost, TelKontakt, WebKontakt, Bankverbindung, Marken.

Jeder Datensatz der Tabellen ist also eindeutig über diesen Index-Wert zu finden.

Zusätzlich gibt es noch einige, in der Grafik nicht dargestellte, Hilfstabellen. Diese dienen lediglich dazu, um häufige Vorgabewerte bequem per Kombinationsfeld im Formular abrufen und damit in den Datensatz einfügen zu können. Beispiel: In der Tabelle **OrtPost** und somit im Formular **Adressen**, gibt es je eine Spalte / ein Feld für das Land (den Staatennamen) und die International Telefonvorwahl. Sicher macht es Sinn, solche festen Werte quasi „vorrätig“ zu haben und nicht jedes mal tippen zu müssen.

So weit zur Tabellenstruktur, kommen wir zum Frontend, also Formularen und Schaltflächen:

Natürlich soll es auch jemandem, (Lieschen Müller) ohne nennen wir es mal „interne Kenntnis“ der Datenverwaltung, möglich sein, mittels der Formulare die Daten richtig und korrekt einzugeben ohne sich hierbei Gedanken darüber machen zu müssen, wie den die Datensätze synchron gehalten werden (siehe Beispiele oben; Mehrere Adressen zu einer Person und umgekehrt, mehrere Mitglieder in einem Verein bzw. ein Mensch der Mitglied in verschiedenen Vereinen ist usw.).

Hierbei soll aber keine Trennung zwischen Eingabe- und Abfrageformularen erfolgen. Vielmehr soll sowohl die Datenerfassung (im Sinne von erster Erfassung) als auch die Dateneingabe (im Sinne von Datenanzeige und nachträglicher Korrektur bzw. Erfassung von zusätzlichen Angaben zur Person, Verein oder Firma), im gleichen Formular erfolgen. Für „reine Abfragen“ im Sinne von z. B. dem Erstellen einer Mitgliederliste, ist die Verwendung der Berichtsfunktion oder auch der Export in verknüpfte OOo-Calc Tabellen, angedacht.

Die Formulare sollen ausnahmslos direkt an die jeweils hinterlegte Tabelle gebunden sein, wobei dies so eigentlich nur für die Hauptformulare gilt, den die Unterformulare greifen ja quasi auf eine Abfrage zu da sie interne über einen SELECT-Befehl verknüpft werden. Gleichwohl glaube ich das es richtig ist, trotzdem die einzelnen Kontrollelemente der Unterformulare (also die Eingabefelder für Werte wie Name, Ort, Geburtsdatum usw.), an die jeweiligen Tabellenfelder zu binden. Intern sind alle Formulare natürlich quasi OOo-Writer Dokumente.

Um das noch mal deutlich zu machen, es geht hier nicht um wechselnde Abfragen (mal eine Mitgliederliste, mal eine Geburtstagsliste, mal eine Adressenliste usw.) über die ein Formular erstellt werden soll. Viel mehr geht es darum, in gleich bleibenden und permanent verfügbaren Unterformularen, einen oder mehrere Datensätze (z. B. Adressen) anzuzeigen bzw. ändern oder hinzu fügen zu können, die aber alle per ID-Wert mit einem, ich nenne es mal Hauptdatensatz (z. B. Person), verknüpft sind. Dazu gibt es, soweit ich bisher verstanden habe, mehrere Möglichkeiten (ResultSet – InsertRow – SQL-Mustervergleich) und ich weiß noch nicht, für welche ich mich entscheiden bzw. wie ich sie konkret anwenden soll.

Aber so weit bisher verstanden, basieren sie alle auf der Methode Formular – Unterformulare. Das diese, ziemlich bequem per Eigenschaftszuweisung mit entsprechenden Auswahlmöglichkeiten wie „Verknüpfung von“ bzw. „Verknüpfung zu“, miteinander zu verbinden sind, ist schon mal sehr nützlich. Da aber die Datenänderung (laut gelesener Literatur) nur bei der direkten Anbindung auch des Formulars an die Tabelle funktionieren soll, bleibt für die deutliche Mehrzahl der Daten (die ja quasi über Abfragen ermittelt bzw. eingegeben werden) nur die Auswahl einer der schon angesprochenen Möglichkeiten.

Bevor ich versuche das Ganze an ein Paar typischen Arbeitsbeispiele deutlicher zu machen, zunächst noch ein Paar Anmerkungen zu den in der grafischen Formulardarstellung verwendeten Kürzeln und Bezeichnungen:

HF steht für eines der 3 Hauptformulare (**Personendaten**, **Clubdaten**, **Firmendaten**). 1x bedeutet einfach, dass diese Formular nur 1 mal vorhanden ist. Was so auch nicht ganz stimmt, denn z. B. Im Formular **Clubdaten** bezieht sich der Button **ZeigeMitglieder** ja wiederum auf die Tabelle **Personendaten**. Dem entsprechend muss es auch ein, wenn auch leicht abgewandeltes, Unterformular für Personendaten geben. Gibt es natürlich auch (es ist quasi das gleiche Formular halt nur ohne Buttons, habe ich nur in der Grafik nicht dargestellt um das nicht noch unübersichtlicher zu machen.

Das, von Clubdaten aus angesteuerte, Unterformular Personendaten heißt dann (entsprechend der verwendeten Bezeichner – zu diesen gleich noch etwas) **mycufPD**.

UF/UF kennzeichnet ein Unterformular, das seinerseits ein Unterformular hat. **UF** kennzeichnet ein reines Unterformular. 2x bzw. 3x dürfte dann klar sein. Die, alle mit **myp...**, **myc...** oder **myf...** beginnenden, Kürzel sind die Bezeichnernamen, über die das jeweilige Formular z. B. in der Basic-IDE angesprochen wird. Er wird einfach über die Eigenschaft „Name“ vergeben. **Tab:** benennt die jeweils hinterlegte Tabelle. Darunter steht dann der eigentlich sichtbare Formularname.

Die Doppellinie in manchen der Formulare bedeutet, das es sich hier quasi um EIN Formular handelt, das Daten aus ZWEI Tabellen anzeigt, dies ist allerdings nur bei diesen, in einer möglichen 1-1 sonst 1-0 Beziehung stehenden, Tabellen der Fall. Tatsächlich handelt es sich natürlich auch um ein Haupt- mit Unterformular. Die werden halt nur quasi (fast) gleichzeitig per Makro geöffnet und bleiben auch permanent mit dem Hauptformular offen.

Die im Formular dunkler dargestellten Felder kennzeichnen Schaltflächen die zur Aktivierung von Makros , besonderes zum Öffnen anderer Formulare, dienen. Die Bezeichnung **mybu...** benennt den Bezeichnernamen (wieder für die Basic-IDE) des Buttons, das darunter stehende Wort bzw. die Wortkombination, ist die sichtbare Beschriftung des Buttons im Formular. Die kleinen Felder mit der Beschriftung **v-r**, **NDS**, **GDS** und **FoS** sind ebenfalls Schaltflächen zum Aufruf eines Makros.

v-r steht für Vorwärts/Rückwärts und dient einfach dazu, um in den angezeigten Datensätzen des Unterformulars den nächsten bzw. vorherigen an zu zeigen. **NDS** steht für Neuen-Datensatz-Speichern, also um in der hinterlegten Tabelle einen INSERT-Befehl aus zu führen. **GDS** steht für Geänderten-Datensatz-Speichern und führt somit in der hinterlegten Tabelle einen UPDATE-Befehl aus. **FoS** Steht für Formular-Schließen, dürfte sich selbst erklären. Entsprechend der bereits angesprochenen Namen für Bezeichner lauten diese dann: **mybuVR**, **mybuND**, **mybuGD** und **mybuFS** oder, falls das notwendig sein sollte (was ich nicht hoffe) auch **mybupND**, **mybucND** usw. usw.

Ich halte diese Schaltflächen für sinnvoll bzw. sogar notwendig, denn in den Unterformularen wird natürlich die Formularsymbolleiste nicht verwendet, somit stehen auch deren entsprechende Schaltflächen nicht zur Verfügung. Nur bei den 3 Hauptformularen wird natürlich die Formularsymbolleiste mit ihren Schaltflächen verwendet, und nur mit deren Vorwärts bzw. Rückwärts Funktion (oder auch per Tab- bzw. Entertaste bei Verlassen des letzten Feldes in der Aktivierungsreihenfolge) soll der angezeigte „Hauptdatensatz“ ausgewählt werden. In den verknüpften „Unterdatensätzen“ wird mit dem (oder vielleicht doch besser den beiden?) Button **v-r** gescrollt. Gerade komme ich darauf das es doch 2 sein müssten. Die könnten dann **dsV** für Vorwärts und **dsR** für Rückwärts heißen, die Bezeichnernamen wären dann also **mybuDSV** bzw. **mybuDSR**. Weil die Grafik aber so weit fertig ist, stelle man sich bitte einfach die beiden an Stelle von **v-r** vor.

Bis zu diesem Punkt lässt sich fast alles (außer der Makroprogrammierung natürlich) mit grafischer Oberfläche und Verwendung von Auswahlmöglichkeiten erledigen. Doch wie kommen die in den Unterformularen eingegebenen bzw. geänderten Daten auch in die richtige Tabelle(n), wie hält man diese dabei auch noch synchron und wie stellt man fest, ob der Datensatz (z. B. die eingegebene Adresse) eventuell sogar schon vorhanden ist? Denn schließlich sind die Tabellen ja mit einem Unique-Index versehen worden um Doppelgänger zu vermeiden. Nun gut, wahrscheinlich kommt dann eine Fehlermeldung schon allein von MySQL, aber das ist ja nicht Sinn der Sache.

Beispiel: Unser „Max Mustermann“ ist schon, auch mit seiner Adresse, in unserer Datenbank erfasst. Auch das er Mitglied im Verein „Die Meistersinger“ ist.

Nun muss der Verein, z. B. aus Kostengründen, sein Vereinsbüro aufgeben.
Und da unser Max Mustermann der Schriftführer ist und er außerdem ein Arbeitszimmer und den leistungsfähigsten PC zu Hause hat, wurde vom Vorstand beschlossen, das seine Adresse nun auch das Vereinsbüro ist und seine Postanschrift auch die des Vereins wird.

Von der Tabellenstruktur aus betrachtet ist auch dieser Fall kein Problem: Die Adresse, also der Datensatz mit seiner **OrtID**, bleibt in der Tabelle **OrtPost** unverändert. Lediglich in der Tabelle **rel_Ort_P_C_F** wäre eine Beziehung zwischen diesem Datensatz (der **OrtID**) und der **ClubID** des Vereins aus der Tabelle **Club** her zu stellen. So weit so gut. Nur wie stellt man das in OOo-Basic an? Nun ja, da wäre eine Dialogbox möglich z. B. mit der Ausgabe:

„Datensatz existiert bereits! In anderem Kontext Speichern?“ und einer Ja/Nein-Auswahl. Nur wie kriegt man das nun wieder hin?
Und wie geht es dann weiter z. B wenn die Auswahl mit „Ja“ beantwortet wird und die anderweitige Speicherung erfolgen soll?

Auf der Suche nach entsprechenden Antworten habe ich zwischenzeitlich einiges gelesen, noch nicht viel davon verstanden, glaube aber mit den nachfolgend beschriebenen Ideen (wobei ich die letzte Frage mit der Ja/Nein Auswahl zunächst mal außen vor lasse) auf dem richtigen Weg zu sein:

Vorher noch eine Bemerkung zu einer angedachten aber dann verworfenen Alternative: Ich hatte in Erwägung gezogen notfalls einfach auch die ID-Schlüsselfelder der Tabellen im jeweiligen Formular/Unterformular sichtbar zu machen und diese, natürlich nur soweit es keine auto-increment Felder sind, sogar zum Schreiben frei zu geben. Damit wäre es wahrscheinlich möglich, die ID-Werte einfach per Drag&Drop zu übergeben. Das hab ich aber wieder verworfen, ist erstens wahrscheinlich ein schlechter Stil und kann zusätzlich, spätestens wenn die ID-Werte auf Grund der großen Datenmenge ein gewisses Maß überschreiten, zu Problemen führen. Und außerdem; Was ist, wenn DAU mal nicht aufpasst – also lass ich das lieber sein.

Bleibt als Alternative wahrscheinlich nur; Sämtliche Dateneingaben (Werte) in allen Formularen, vielleicht mit Ausnahme der 3 Hauptformulare (weil die ja direkt an die Tabellen gebunden sind), zunächst in einen Zwischenspeicher zu bringen, dessen Inhalt mit dem der Tabelle zu vergleichen und dann, je nach Ergebnis des Vergleichs, in die Tabelle zu speichern. Womit ich beim Thema Variablen bzw. Arrays ankomme.

Soweit bisher verstanden, sind sowohl Variablen als auch Arrays als solche Zwischenspeicher geeignet. Der wesentliche Unterschied besteht darin, das eine Variable, mit Ausnahme des Typs „Variant“ von dessen Verwendung aber (laut Literatur) abgeraten wird, nur Daten eines Typs (z.B. Text oder Datum) aufnehmen kann.

Nun will ich aber nicht nur einen einzelnen Wert (z. B. Name ODER Geburtsdatum) bei der Eingabe vergleichbar haben, sondern den gesamten Datensatz bzw. besser gesagt, den Teil des Datensatzes der, über das Formular, in die hinterlegte Tabelle soll. Also bei einer Person z. B. Anrede, Vorname, Familienname, Rufname, Geburtstag usw. Ob ich da mit Variablen weiter komme, weiß ich gar nicht so recht denn, soweit bisher verstanden, nimmt eine Variable ohnehin nur EINEN Wert auf.

Nun trifft zwar in einigen der Tabellen zu, das alle Felder (sind alle varchar) nur Text aufnehmen, in anderen kommen aber auch Datumsfelder (date) und Textfelder (varchar) bzw. Memofelder (text) gemischt vor. Hinzu kommt, das ja ggf. auch der ID-Wert (integerzahl, in der Variable dann als long) verglichen werden soll bzw. muss. Oder ist dies (ID-Werte vergleichen) gar nicht notwendig weil ich ja die Verknüpfung der Tabellen und somit der Formulare, bereits über die ID-Werte per SELECT-Befehl vornehme? Oder sollte der ID-Wert zwar verglichen werden, aber außerhalb der Variable / des Arrays?

Das hab ich nun vom „Schlaue Bücher lesen“, fast so viele Fragen wie Antworten. Zumindest habe ich verstanden, das auch diese Objekte (Variable bzw. Array) einen eindeutigen Namen zu bekommen haben. Sie heißen dann z. B. für das Formular Personendaten und die entsprechende Tabelle **myavPD**, für Adressen **myavOP**, für BezugZumOrt **myavROPCF** usw. Nach **myav** verwende ich hier einfach eine sinnvolle Abkürzung der hinterlegten Tabelle.

Im übrigen bin ich gerade zu der (hoffentlich richtigen Erkenntnis) gekommen, das wahrscheinlich die Verwendung einfacher, eindimensionaler Arrays, mit der entsprechenden Indexzahl je nach Anzahl der Tabellenfelder deklariert, in meinem Fall das richtige wäre.

Frage1: Ist das so? Wenn ja, wie deklariere ich die richtig, besonderes im Bezug auf die unterschiedlichen Wertetypen (Text / Datum)?

Wenn nein, wie macht man das besser?
Ich denke mal, das war genug Theorie. An praktischen Arbeitsbeispielen wird das ganze vielleicht etwas deutlicher:

Nun sitzt also unser „Lieschen Müller“, wie die meisten Leute nur ein wenig mit Word und Excel bekannt, vor dem PC, hat das Anmeldeformular vor sich auf dem Desktop und gerade mal eine kurze Einweisung in die Datenbank sowie ihr ein Passwort bekommen.

Nach dem Anmelden mit dem richtigen Passwort (sonst Dialogbox mit Fehlermeldung, wie ich das ganze Ding / Anmeldeformular gebastelt kriege, ist mir auch noch nicht ganz klar, wäre **Frage2**), kann sie einen der drei sichtbaren Button (werden erst dann frei gegeben) auslösen. Sie will Daten einer Person erfassen und klickt den entsprechenden Button (ich verwende im folgenden nur die Bezeichnernamen für die IDE) **mybuAFpd** und löst damit ein Makro aus, das nenne ich mal: **MAKophfPD**.

Diese **MAKopfoPD** tut folgendes:

1. Öffnet **myphfPD**, 2. Öffnet quasi gleichzeitig **mypufPP** und zeigt 3. den ersten Datensatz in der/den hinterlegten Tabelle(n) an. Der, bereits bei der Einrichtung der Formulare dafür definierte, SQL-Befehl (die schreibe ich folgenden so, wie ich sie laut Buch von Kofler für richtig halte) sieht so aus:

– SELECT * FROM **Person, PersProfil** WHERE **Person.PersID** = **PersProfil.PersID** ORDER BY Name, Vorname;

Nach Eingabe der entsprechenden Daten werden diese, z. B. durch anklicken des Buttons **NDS** im Formular, in der/den Tabelle(n) gespeichert. Bereits an dieser Stelle taucht eine weitere Frage auf, denn:

Die eingegebenen Daten wären ja, soweit auch Daten im Unterformular **mypufPP** eingegeben wurden (was nicht sein muss, sofern die nicht bekannt sind), quasi durch EIN mal drücke/anklicken von **NDS** in ZWEI Tabellen zu speichern. OK, man kann natürlich auch erst den Speichern-Button in der ja vorhandenen Leiste des Hauptformulars betätigen und danach **NDS** im Unterformular. Das halte ich aber nicht für sonderlich elegant. Außerdem möchte ich INSERT bzw. UPDATE Befehle über zwei oder mehr Tabellen vermeiden und auch die Arrays will ich ja nur über ein Formular bzw. die Werte der hinterlegten Tabelle, definieren. Möglicherweise lässt sich das so lösen:

Einfach dem letzte Datenfeld von **myphfPD** in der Aktivierungsreihenfolge, bei Fokusverlust, einen INSERT-Befehl zuweisen. Nur was mach ich dann, wenn es eigentlich nur ein UPDATE-Befehl hätte sein müssen? Oder verwende ich an der Stelle besser grundsätzlich einen UPDATE-Befehl im Zusammenspiel mit einer IF – THEN Prozedur? Das ganze ist nun **Frage3**.

Aber dazwischen liegt ja noch die ganze Vergleichsprozedur also Array deklarieren, Werte zuweisen, diese Werte mit Tabelle vergleichen. Ich vermute mal, dass ich das Ganze (schon allein wegen der Lebensdauer der Arrayvariable) in einem Makro abhandeln muss. Das **MAKophfPD** wird ja dann schon fast eine kleinere Anwendung. Oder kann ich z. B. dem Ereignis „Modifiziert“ in den Kontrollfeldeigenschaften, oder vielleicht besser doch dem Ereignis „Vor der Datensatzaktion“ in den Formulareigenschaften, ein entsprechendes Makro (das hieße dann z. B. **MAKdatacheck**) zuweisen und in diesem die Prozeduren bis zum Speichern, also bis zum INSERT- bzw. UPDATE Befehl der ja **mybuND** bzw. **mybuGD** zugewiesen wird, ablaufen lassen?
Wie ich das am besten anstelle, wäre **Frage4**.

Lieschen Müller hat nun also unseren „Max Mustermann“ erfasst, jetzt will sie noch seine Adresse eingeben. Nehmen wir dafür mal an, er hat zwei davon, die erste (an der er auch gemeldet ist) quasi zu Hause bei den Eltern, die zweite (an der er tatsächlich mehr oder weniger wohnt, ohne dort gemeldet zu sein) bei seiner Partnerin im Nachbarort. Lieschen Müller klickt also auf den Button **mybupBA** und löst damit folgende Aktionen aus; Das Makro **MAKopufAD** wird ausgeführt und erledigt folgende Arbeitsschritte:

1. Öffnet **mypufBO**, 2. Öffnet quasi gleichzeitig **mypufAD** und auch 3. **mypufOI** und zeigt 4. den ersten Datensatz in der/den hinterlegten Tabelle(n) an. Der, bereits bei der Einrichtung der Formulare dafür definierte, SQL-Befehl (wieder laut Buch von Kofler) sieht so aus:

– SELECT * FROM **Person, rel_Ort_P_C_F, OrtPost, OrtInfo** WHERE **Person.PersID = rel_ort_P_C_F.PersID** AND **OrtPost.OrtID = rel_Ort_P_C_F.OrtID** AND **OrtPost.OrtID = OrtInfo.OrtID;**

Aus der Einweisung weiß unser Lieschen Müller, das sie jeden Datensatz speichern muss bevor sie den nächsten eingibt. Sie klickt also in den Formularen **mypufBO** und **mypufAD** (und ggf. auch im Formular **mypufOI**) auf den Button **NDS** und schließt damit die Eingaben ab. Jetzt steht aber noch die zweite Anschrift von „Max Mustermann“ auf der Karteikarte, auch mit dem Hinweis, das ist die Adresse der Freundin.

Lieschen Müller klickt also per Button **dsV** im Formular **mypufBO** zum nächsten (noch leeren) Datensatz, im einem Auswahlfeld/Kombinationsfeld für Vorgabewerte (aus einer der schon angesprochenen Hilfstabellen) wählt sie z. B. „Zweitwohnung bei Partner/in“ aus, wechselt weiter zum Formular **mypufAD** und gibt dort die Adresse ein. Als sie dann wieder zum Speichern auf **NDS** klickt passiert aber folgendes:

Es öffnet sich die schon angesprochene Dialogbox mit der Meldung: **Datensatz existiert bereits! In anderem Kontext Speichern?**

Grund dafür ist der unique-index in der Tabelle **OrtPost** der ja Doppelgänger vermeiden soll. Natürlich will sie die Adresse speichern und beantwortet den Dialog entsprechend mit JA. Damit das dann auch richtig passiert, muss ja quasi nur in der Tabelle **rel_Ort_P_C_F** die Beziehung zu „Max Mustermann“ hergestellt werden. **Frage5** lautet, kann das so funktionieren:

INSERT INTO **rel_Ort_P_C_F.OrtID** SELECT **OrtPost.OrtID**; und diesen Befehl einfach der JA-Antwort zuweisen?

Damit sollte theoretisch die **OrtID** der „doppelten“ Adresse ins richtige Feld **OrtID** der Tabelle **rel_Ort_P_C_F** geschrieben sein.

Doch was ist mit der „anderen Seite“ (also der **PersID** von „Max Mustermann“) in der Tabelle **rel_Ort_P_C_F**. Wegen dem zusammengesetzten Primär-schlüssel der Tabelle kommt natürlich auto-increment nicht in Betracht und somit wird natürlich auch kein Wert in die Tabelle übernommen. Also muss ich diesen (**PersID** für **rel_Ort_P_C_F**) wohl auch erst per SELECT Befehl holen also wieder per z. B.:

INSERT INTO **rel_Ort_P_C_F.PersID** SELECT **Person.PersID**;

Oder würde sich dieser Schritt, quasi bereits durch die erfolgte Formularverknüpfung per SQL-Befehl, bzw. die Eigenschaftszuweisung „Verknüpfung Von“ bzw. „Verknüpfung Zu“, entfallen können. Der Punkt wäre dann **Frage6**.

Bevor ich mir weiter denn Kopf zerbreche, hoffe ich einfach mal, die ganze Sache so halbwegs verständlich dargestellt zu haben.

Wenn sich jetzt noch jemand findet der sich mit der Materie auskennt, dies liest, sich ein paar Gedanken dazu macht und mir diese auch noch mitteilt, dann bringt das wahrscheinlich nicht nur mich weiter. Ich bin jedenfalls für alle Anregungen, Hinweise und Vorschläge dankbar.