GEDCOM-Standards 5.5/5.5.1/5.5.5 kritisch betrachtet und verständlich erklärt

Inhalt:

1 Ein wenig Theorie 1

1.1 Was ist GEDCOM und was nicht? 1

1.2 Was ist GEDCOM wirklich? 2

1.3 Wofür wird GEDCOM eingesetzt? Der Stammbaum 3

1.3.1 Biologische Vererbung 3

1.3.2 Juristische Vererbung 3

1.3.3 Paarbeziehung/Ehe 4

2 GEDCOM-Struktur 4

2.1 Symbole 4

2.2 Datenmodell 5

2.2.1 Physische Elemente 5

2.2.2 Virtuelle Elemente für die Beziehungen 7

2.2.3 GEDCOM 5.5.5 7

2.2.4 GEDCOM EL 8

2.3 Grammatik zur Darstellung der Daten 8

2.3.1 Konzept 8

2.3.2 Grammatikregeln 9

2.3.3 Grammatiksyntax 10

2.3.4 Redundanz und Widerspruch (referenzielle Integrität) 10

2.3.5 Problem: Huhn oder Ei? 11

2.4 Verwaltungsdaten 11

2.4.1 Kopf 11

2.4.2 Einreichung 12

2.4.3 Abschluss 12

2.5 Tabellen 12

2.5.1 Person (individual) 13

2.5.2 Familie (family) 14

2.5.3 Medien (multimedia) 15

2.5.4 Anmerkungen (notes) 16

2.5.5 Quellarchiv (repository) 16

2.5.6 Quelle (source) ... 17
2.5.7 Einreichung (submission), Verfasser (submitter) ... 17
2.6 Nachschlagtabellen (Unterstrukturen) ... 18
2.6.1 GEDCOM-Unterstrukturen ... 18
2.6.2 Weitere Nachschlagtabellen ... 18
2.7 Zusammenfassungen ... 19
2.8 Zeichensatz und Verschlüsselung ... 19
2.8.1 ASCII ... 20
2.8.2 ANSEL ... 20
2.8.3 UNICODE ... 20
2.8.4 Konvertierung ... 21
2.8.5 BOM ... 21
2.8.6 Datum und Datumsformat ... 22
2.8.7 UTF-8 ... 23
2.8.8 Zeilenschaltungen ... 23
2.8.9 Leerraum ... 23
2.8.10 Fortsetzung mit CONC/CONT ... 23
2.8.11 Anmerkungen ... 24
2.8.12 Nichttransparente Zeichen ... 24

3 Flache Tabellen ... 25
3.1 Schlüssel (Primär-, Fremdschlüssel, Verweise) ... 25
3.2 Eingabe ... 25

4 Datenbankentwurf ... 25
4.1 Strukturvorschlag ... 25
4.2 Invarianz und Historie ... 27
4.3 Ereignisorientierter Entwurf ... 27
4.4 Kernentwurf I ... 29
4.4.1 Allgemeiner Beziehungstyp ... 29
4.4.2 Ereignisse vs. Zustand ... 31
4.4.3 (A)Symmetrie ... 31
4.4.4 Reflexive Beziehungstypen ... 32
4.4.5 GEDCOM und die freien Beziehungstypen ... 32
4.5 Kernentwurf II ... 32
4.5.1 GEDCOM und die festen Beziehungstypen ... 33
4.6 Erweiterungen ... 33

4.7 Was kann GEDCOM abbilden? ... 33
4.8 Ausblick auf GEDCOM X ... 33
4.8.1 Datentypen der obersten Ebene (Top-Level Data Types) ... 34
4.8.2 Nachschlagtabellen (Component-Level Data Types) ... 35
4.8.3 GEDCOM X Beziehung (relationship) ... 36

5 GEDCOM und RDBM ... 37
5.1 Normalformen ... 37
5.2 Transformationsregeln ... 37
5.3 Beurteilung ... 38

6 GEDCOM-Datenaustausch ... 39
6.1 Informationsverluste ... 39
6.2 Test ... 39
6.2.1 Selbsttest ... 39
6.2.2 Fremdtest ... 40
6.3 Validation ... 40
6.3.1 Formale Validierung ... 40
6.3.2 Hilfsweise (manuelle) Validierung ... 40
6.3.3 Logische (semantische) Validierung ... 41

7 GEDCOM-Dateien zusammenführen ... 41
7.1 Gleichheiten erkennen ... 41
7.2 Ähnlichkeit erkennen ... 41
7.2.1 Soundex ... 42
7.2.2 NYSIIS (New York State Identification and Intelligence System) ... 42
7.2.3 Soundex für die deutsche Sprache ... 43
7.2.4 Kölner Verfahren ... 43
7.2.5 Methaphone ... 44
7.2.6 Silbentrennung ... 44
7.2.7 Übereinstimmung ... 44
7.3 Zusammenführen ... 45
7.3.1 Eigenschaften ... 46
7.3.2 x:mx-Beziehungstypen ... 46
7.3.3 mx:mx-Beziehungstypen ... 46

8 GEDCOM und Online-Datenbanken ... 46
8.1 GEDCOM und GEDBAS ... 47
8.1.1 Quellenangaben ... 47

8.1.2 Ortsangaben 47
8.1.3 Nachname/Geburtsname/Suchname 47
8.1.4 Vorname/Rufname und Zusätze 48
8.1.5 Sortierung 48

9 Auswertungen 49
9.1 Data Pipeline 49

10 Realisation in Gramps 50
10.1 Datenmodell 50
10.1.1 Tabellen 50

wurde am 02.10.2019 auf folgender Seite veröffentlicht:

https://www.gedcom.org/gedcom.html

Dies Spezifikation umfasst 166 Seiten, die hier natürlich nicht vollständig im Inhalt und im Vergleich zu den Vorgängerversionen dargestellt werden können.

1 Ein wenig Theorie

1.1 Was ist GEDCOM und was nicht?

GEDCOM ist ein **Datenaustauschformat** ähnlich XML (Extended Markup Language), mit dem sich genealogische Daten zwischen zwei Programmen (Sender/Exporteur und Empfänger/Importeur) in Form einer Textdatei austauschen lassen. Ein „wissender“ Leser kann die Daten interpretieren. Etwas profaner ausgedrückt ist es ein Dateiformat auf Textbasis.

Obwohl die letzte offiziell verabschiedete Version 5.5 aus dem Jahr 1996 stammt und die Version 5.5.1 (Oktober 1999) nie verabschiedet – aber stillschweigend genutzt – und als Version 5.6 bzw. 6.0 2001 sogar abgelehnt wurde, hat sich das Format quasi als weltweiter Standard etabliert. Organisationen, die GEDCOM verbessern wollen,

https://fhiso.org/
http://www.gedcomx.org/ mit den Spezifikationen http://www.gedcomx.org/Specifications.html

kommen und gehen wie BetterGedcom und die große Dunkelziffer weiterer Seiten.

Da FamilySearch die Version 5.5.1 seit der Veröffentlichung von PAF 5.0 im Jahre 2000 selbst benutzt, ist die Version 5.5.1 nicht mehr als Entwurf zu werten. Einen Überblick über die Versionen findet man auf

https://www.gedcom.org/versions.html

Hier wird die Version 5.5.5 mit „Wartungsfreigabe. Qualität. Einfacher und strenger“ tituliert. Offensichtlich fehlen einige Versionen, die nie veröffentlicht wurden (oder die es nie gab).

GEDCOM ist **keine Datenbank**. Ein **Datenbankmanagementsystem (DBMS)** besteht aus der Datenhaltung (Datenbasis) und der der Verwaltungssoftware. Wesentliche Anforderungen an ein DBMS sind

- Mehrbenutzerfähigkeit (multi user capability) (Abb. 1)
- Bestätigte Speicherung

GEDCOM ist auch kein **Kommunikationsprotokoll**, wie es das COM im Namen vermuten lässt, vergleichbar mit http. GEDCOM ist ein Dateiformat, das die Struktur einer Datei zum Datenaustausch zwischen Programmen definiert.

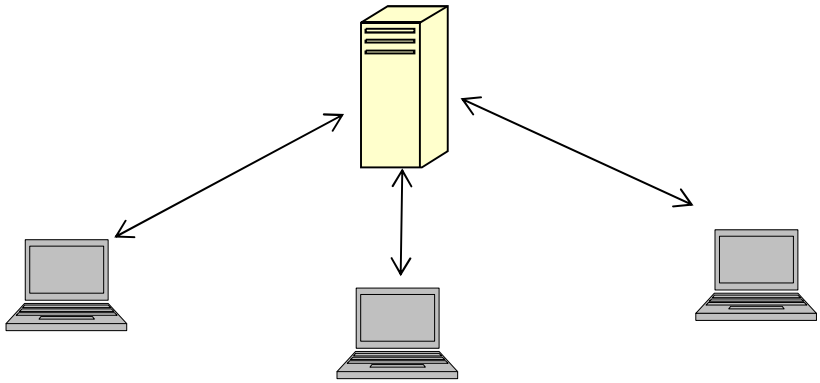


Abb. 1: Klient-Server-Architektur

Beides wird durch GEDCOM-basierte Programme (z. B. Ages) nicht erfüllt. Typischerweise besteht der Arbeitsablauf der Programme ohne eigene Datenhaltung aus:

- GEDCOM-Daten in ein Einplatz-Programm einlesen
- Daten intern als Objekte verarbeiten
- Daten im GEDCOM-Format abspeichern

Auch viele Programme mit eigener Datenhaltung (Ahnenblatt usw.) sind nicht mehrbenutzerfähig, auch wenn sie die eingegebenen Daten sofort dauerhaft speichern.

Internet-Ahnenprogramme müssen dagegen immer (zumindest lesend) mehrbenutzerfähig sein.

Eine GEDCOM-Datei ist eine **Textdatei**, keine Binärdatei. Sie ist wie ein Textdokument in unterschiedlich langen Zeilen strukturiert, die ein Mensch lesen aber nur schwer schreiben kann.

Basis der folgenden Betrachtungen ist die deutsche Übersetzung der GEDCOM Version 5.5.1 (08.05.2012) von

http://www.daubnet.com/de/gedcom

sowie die Spezifikation der verschiedenen (neueren) Versionen auf

https://www.gedcom.org/gedcom.html

Die noch ältere Version 5.5 im Original

http://homepages.rootsweb.com/~pmcbride/gedcom/55gctoc.htm

Hier gibt es einige Abweichungen, z. B. bei der Beschreibung der Multimedia-Dateien.

1.2 Was ist GEDCOM wirklich?

GEDCOM ist eine Entwicklung der „The Church of Jesus Christ of Latter-day Saints“ (LDS) aus dem Jahre 1984. Es ist daher ein *religiöses* Format, das dazu dient, das religiöse Modell der LDS als Datenbank zu modellieren. LDS ist die umfangreichste Ausprägung des Mormonen-Kultes. Durch das Zusammentragen aller Ahnen erreichen auch diese das Himmelreich der Mormonen. Daher stehen beispielsweise Geburt und Kinds- bzw. Erwachsenentaufe im Mittelpunkt, während der Tod anfänglich von den Mormonen gar nicht erfasst wurde.

Für die Standardisierung war das **Family History System** (FHS) zuständig, Dieses hat sich inzwischen in **FamilySearch** umbenannt. Die Organisation ist aber seit den (gescheiterten) Vorarbeiten zur Version 5.5.1 quasi in den Ruhestand gegangen, da die Version 5.5 offensichtlich alle für die LDS notwendigen Daten perfekt abbildet und daher nicht mehr verändert werden muss. Seit dem 02.10.2019 enthält die neu gestaltete Seite

https://www.gedcom.org/index.html

eine neue Version 5.5.5 mit Anmerkungen.

Grundsätzlich ist z. B. die Familie (Ehe) die einzige erlaubt Form des Zusammenlebens zweier (getrennt geschlechtlicher) Personen, was aber durch die Praxis inzwischen überholt ist.

GEDCOM besteht im Grunde aus zwei Spezifikationen. Die Basis bildet das GEDCOM data format, das (ähnlich XML), welches das Grundformat und die Syntax festlegt. Die darüber liegende Ebene definiert ein darauf aufbauendes Schema (Lineage-Linked Form) ähnlich der XML Document Type Definition (DTD).

1.3 Wofür wird GEDCOM eingesetzt? Der Stammbaum

Ein Stammbaum stellt die Abstammung eines Objekts (Lebewesen, Sache, Idee) (Kindobjekt) von möglicherweise bis zu zwei weiteren Objekten (Elternobjekte, Vorgängerobjekte) dar.

Ein Stammbaum ist kein mathematischer (binärer) Baum, der verlangt, dass die Elternobjekte alle unterschiedlich sind. Durch Ahnenschwund können zwei Kinder innerhalb des Stammbaumes gleiche Vorgängerobjekte besitzen. Es entsteht die **Ahnentafel**. Mathematisch ist sie ein *gerichteter, azyklischer Graph*. Normalerweise ist die Ahnentafel *zusammenhängend*, d. h., hat nur eine einzige Wurzel. Da wir die Knoten auch mit Personen belegen, ist die Ahnentafel *verzweigt*.

1.3.1 Biologische Vererbung

Meist gehen wir davon aus, dass ein Kind zwei biologische Eltern hat, die durch Verschmelzung von Ei- und Samenzelle das Kind erzeugen. Beide Zelltypen enthalten den halben Chromosomensatz der durch die Verschmelzungen wieder zu einem vollständigen Chromosomensatz wird.

X- und Y-Chromosomen bestimmen das Geschlecht. Das Y-Chromosom wird nur durch die Männer vererbt. Daher hat (hatte) die männliche Stammlinie eine besondere Bedeutung.

Heute wissen wir, dass die Mitochondrien der Eizelle nur durch die Frauen vererbt werden. Es entsteht eine weibliche Stammlinie.

Störungen können bei der Auftrennung der Chromosomensätze zur Ei- bzw. Samenzelle auftreten, so dass es beispielsweise Verdoppelungen einzelner Chromosomen kommt. So kann das juristische Geschlecht vom biologischen abweisen.

Inzwischen sind menschgemachte Eingriffe bekannt, indem die Eizelle einer dritten Frau entkernt und mit dem halben Chromosomensatz der (zukünftigen) Mutter sowie dem Vater beschickt wird. Somit hat das Kind zwei biologische Mütter und einen biologischen Vater.

Kuckuckskind bezeichnet ein Kind, dessen Vater nicht sein biologischer Vater ist, weil die Mutter es mit einem anderen Mann zeugte und das Kind und seinen sozialen Vater im Glauben ließ, miteinander blutsverwandt zu sein.

Ein Nachweis der Blutsverwandtschaft ist in der Vergangenheit außer durch Offenbarung der Mutter („auf dem Sterbebett“) unmöglich. Aber auch heutzutage gibt es immer noch juristische Hindernisse zum Nachweis der biologischen Vaterschaft.

Über die Anzahl der Kuckuckskinder gibt es unterschiedliche Annahmen zwischen 10 % und 30 %. Neuere Untersuchungen gehen von einer weitgehend geringeren Anzahl von etwa 1 % aus:

https://www.forschung-und-wissen.de/nachrichten/psychologie/es-gibt-weniger-kuckuckskinder-als-angenommen-13372389

Inzwischen gibt es einen Hype, um per DNS-Analyse seine Vorfahren zu bestimmen.

https://www.heise.de/newsticker/meldung/US-Ermittler-sollen-Gendaten-fuer-inverse-Ahnenforschung-nutzen-4541458.html

Auch wenn wir die Entscheidung treffen, dieses Verfahren zu nutzen, da wir ja zu den „Guten“ zählen und „nichts zu verbergen haben“, sollten wir uns darüber klar sein, dass wir eine Entscheidung treffen, die Nebenwirkungen für unsere Verwandten, unsere Kinder und Kindeskinder hat, denen wir unsere DNS weitergeben. Wir greifen damit in die Rechte dieser Personen ein, nur um eine Verwandte dritten Grades auf dieser Welt zu finden, die eigentlich durch die Sperrfristen bei den Personenstandsbüchern geschützt sein sollte.

1.3.2 Juristische Vererbung

Das Hauptproblem bei der Aufstellung einer Ahnentafel besteht darin, dass es neben der biologischen Vererbung auch noch eine juristische Vererbung gibt, die nicht immer übereinstimmen, trotzdem will der Ahnenforscher (Genealoge) sie möglichst mit einem einzigen Programm abbilden.

Durch **Adoption** kann ein Kind (beliebig) viele weitere Eltern erhalten (wenn auch mit der Einschränkung der zeitlichen Abfolge).

Der ursprüngliche Ansatz von GEDCOM (bis zur Version 5.5.1) bestand in der idealen Familie auf den Eltern und ihren Kindern. GEDCOM 5.5.5 geht von der in den Dokumenten niedergelegten (juristischen) Vererbung aus, die nicht unbedingt die biologische sein muss.

1.3.3 Paarbeziehung/Ehe

Die letzte Aussage führt uns fast nahtlos zu einem anderen Problem. GEDCOM 5.5.1 geht von einer getrennt-geschlechtlichen Verbindung aus, wobei es pro Familie einen Ehemann (`HUSB`) und eine Ehefrau (`WIFE`) gibt, deren unterschiedliches Geschlecht damit implizit festlegt.

Die Realität liefert uns aber inzwischen gleichgeschlechtliche Verbindungen. Schon wird es schwierig, welche der beteiligten Personen der Ehemann und welche die Ehefrau ist. Da GEDCOM Zeiger in beiden Richtungen (`FAM.HUSB -> INDI, FAM.WIFE -> INDI; INDI -> FAM`) verwaltet, sollte es keine Probleme bei der Verarbeitung geben.

Das RDM hat mit der Suche Probleme z. B. bei der Suche des Partners, da der Schlüssel einer gefundenen Person nun sowohl in `FAM.HUSB` als auch in `FAM.WIFE` auftreten kann. Hier rettet uns das Modell der allgemeinen Beziehung mit einem Typ `Geschlecht`.

Besonders in dünn besiedelten, schwer zugänglichen Gebieten kommt es zu Ehen zwischen nahen Verwandten. In den Kirchenbüchern finden wir dazu spezielle Freigabevermerke. Oft sind dann der Geburts- und Ehename gleich. Programmtechnisch handelt es sich natürlich um zwei getrennte Personen. Auf dem Bildschirm sehen sie aber gleich aus.

Wie gehen wir mit Mehrfachehen um? Einige Religionen/Staaten lassen es zu, dass ein Mann mehrere Ehefrauen hat. Jede Verbindung muss separat erfasst werden.

Wenn dies juristisch ausgeschlossen ist, dann treten mehrere Ehen einer Person nur zeitlich hintereinander auf. Eine Überprüfung auf Zulässigkeit kann aber nur dann geschehen, wenn das Ende der vorherigen Ehe (Partnertod, Scheidung) festgestellt ist.

In ganz seltenen Fällen kommt es sogar vor, dass sich Paare trennen und später wieder heiraten (was bei Schauspielern publik wurde). Weitere Diskussionen finden wir in den Artikeln auf

https://www.tamurajones.net/articles.xhtml

2 GEDCOM-Struktur

2.1 Symbole

Um den GEDCOM-Standard zu verstehen, müssen wir auf folgende Symbole achten:

Symbol	Erklärung
`Element := Details`	Element „besteht aus“ den folgenden Details. Element „ist definiert als“ die Details. Eine Technik, um formale Sprachen zu definieren.
`<<Struktur>>`	Verweis auf eine untergeordnete (weiter gegliederte) GEDCOM-Struktur, die hier eingesetzt werden kann (Nichtterminalsymbol, Variable).
`<Primitiv>`	Wert eines primitiven (nicht weiter zerlegten) Elements (Terminalsymbol).
`{Min:Max}`	Minimale und maximale Anzahl von Wiederholungen eines Elements relativ zum übergeordneten Element (wenn das Element fehlt, dann fehlen auch alle seine Kindelemente).
`[Optionen]`	Liste der möglichen Optionen, optional bei Einzelelemente.
`+`	Verbinder zwischen zwei aufeinanderfolgenden Elementen
`|`	Trenner in der Liste der möglichen Optionen (:= oder).
`n`	Ebenennummer
`+1,+2,…`	Ebenennummer erhöht sich entsprechend (jedoch ohne Lücken) relativ zum Basiselement.
`0xhhh`	Die Angabe erfolgt hexadezimal.

In Klammern stehen einige Begriffe aus der Theorie der formalen Sprachen.

Die Spezifikationen der Versionen 5.5 und 5.5.1 sind an vielen Stellen formal nicht immer eindeutig. So werden die Begriffe `record`, `subrecord` und `structure` gern unklar benutzt.

2.2 Datenmodell

Da Programme wie AGES darauf verweisen, das Datenmodell von GEDCOM zu realisieren, sollten wir es näher untersuchen.

GEDCOM nennt sein Datenmodell **lineage-linked form** (abstammungsverknüpftes Schema), ein (ehe-) paarzentrierter Ansatz, bei dem das Ehepaar und seine Kinder als wichtigstes Element erscheinen. Dabei werden die Eltern mit ihren Kindern über mehrere Generationen hinweg in Beziehung gesetzt, so dass der Stammbaum entsteht. In der Praxis gibt es aber eine Vielzahl von Abweichungen bei diesem Idealmodell. Einmal abgesehen von allen Sonderfällen stoßen wir irgendwann in unserem Stammbaum auf Personen, deren Eltern wir nicht kennen, die aber trotzdem verwandt sind (Geschwister). Nun kann man streiten: Woran erkennt man Geschwister? Natürlich daran, dass sie gemeinsame (derzeit noch unbekannte) Eltern besitzen.

2.2.1 Physische Elemente

Auch wenn GEDCOM selbst keine Datenbank ist, so liegt ihm ein fiktives Datenmodell zugrunde. Ein Datenmodell ist eine Abbildung der realen Welt auf Bits und Bytes. Dieses Modell soll nun zumindest grob entwickelt werden.

Am einfachsten ist es, von tatsächlich vorhanden, physischen Elementen auszugehen. Dies sind statische Elemente, die der Forscher in Tabellen, Ordnern usw. sammelt, also „anfassen“ kann. Wenn sich der Ahnenforscher umschaut, dann findet er hauptsächlich

- Personen (Individuen, individuals), die ihn umgeben
- Dokumente (Quellen, sources), die auf seinem Schreibtisch liegen
- Bilder, Filme (Multimedia), obwohl man auch die Dokumente dazu zählen könnte
- Orte (places, locations), die er besuchen kann
- Anmerkungen, Notizen zu den vorherigen Elementen
- Verwaltungsnotizen (Was habe ich schon, was muss ich noch erledigen?)

Was finden wir dazu in GEDCOM?

Das Datenmodell versteckt sich in den Haupt-Datensätzen (top-level records). Im RDM *Entitätstypen* oder einfacher den Zeilen einer *Tabellen* genannt. Die Menge der erlaubten Haupt-Datensätze sind in der Norm mit der Kardinalität `{1:1}` aufgezählt und durch Oder | verknüpft:

```
RECORD :=
   [
    n <<FAM_RECORD>>                    {1:1}
    |
    n <<INDIVIDUAL_RECORD>>             {1:1}
    |
    n <<MULTIMEDIA_RECORD>>             {1:1}
    |
    n <<NOTE_RECORD>>                   {1:1}
    |
    n <<REPOSITORY_RECORD>>             {1:1}
    |
    n <<SOURCE_RECORD>>                 {1:1}
    |
    n <<SUBMITTER_REPOSITORY>>
   ]
```

Wobei n die aktuelle Ebenennummer ist (Hauptdatensätze beginnen immer mit der 0).

In den Haupt-Datensätzen befinden sich folgende Daten:

Record	Tag	Inhalt	Anmerkung
INDIVIDUAL-RECORD	INDI	Personendaten	Einzelne Person
FAMILY-RECORD	FAM	Familiendaten	Fiktive Familie
NOTE-RECORD	NOTE	Anmerkungsdaten	

Record	Tag	Inhalt	Anmerkung
REPOSITORY-RECORD	REPO	Quellarchivdatensatz	Archiv, in der die Quelle vorliegt
SOURCE-RECORD	SOUR	Quellendaten	Bibliografische Quellenbeschreibung
SUBMITTER-RECORD	SUBM	Einreicherdaten	Mussfeld im Kopf: Einreicher sonst Kannfeld: Verfasser

Jedes Element wird durch einen (nicht global eindeutigen) Tag (Kürzel, Kennung) beschrieben. Eine Sammlung aller Tags über die Versionen hinaus finden wir im Internet auf der Seite:

https://www.tamurajones.net/GEDCOMTags.xhtml

Die Bezeichnung der in GEDCOM erlaubte Deklaration von benutzereigenen Tags beginnt mit einem Unterstrich (wie in alten C-Zeiten).

Ein Record kann selbst wieder Records enthalten, die so genannten Subrecords. GEDCOM 5.5.5. bezeichnet jede Zeile in einer GEDCOM-Datei als Record und weicht damit von der Bezeichnung in Datenbanken ab, die von Feldern/Spalten/Eigenschaften sprechen. Damit sind gleichnamige Tags kontextabhängig und müssen bei Bedarf eindeutig qualifiziert werden, indem wir die Hierarchie durch Punkte getrennt aufzählen.

In der Liste der Haupt-Datensätze (Tabellen, Abb. 2) erkennen wir Zusatztabellen wie `Submitter` für die Überspielung an die Mormonen-Datenbank, die für eine private Ahnenverwaltung nicht unbedingt notwendig sind. Andererseits sehen wir eine wesentliche Tabelle nicht:

Es gibt keine Ortsdatensätze.

Dieser Fakt wird in GEDCOM 5.5.5 explizit als Design-Fehler von GEDCOM 5.5.1 herausgestellt (S. 72).

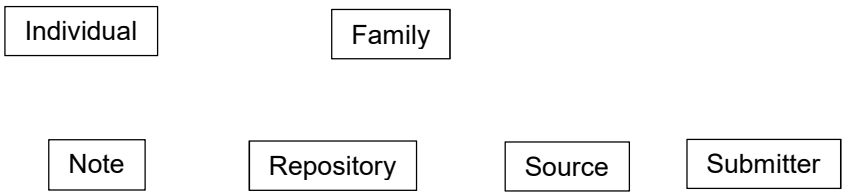


Abb. 2: Datensätze (Tabellen) in GEDCOM

(Geburts-, Tauf-, Einsegnungs-, Heirat-, Scheidungs-, Todes-, Begräbnis- usw.) Orte für ganze Sippen müssen also immer wieder, ggf. redundant, eingegeben werden. Die `<<PLACE_STRUCTURE>>` wurde in GEDCOM 5.5.1 neu in die `<<EVENT_DETAIL>>` Struktur aufgenommen, so dass für jedes Ereignis einige wenige Zusatzinformationen zu einem Ort gespeichert werden können. In GEDCOM 5.5.1 gab es das `<PLACE_HIERARCHY>` Element, um die juristische Unterstellung eines Ortes darzustellen, eine durch Kommata getrennte, textuelle Aufzählung von juristischen Organisationsnamen. Leider kann sich diese Hierarchie für einen konkreten Ort im Laufe der Zeit ändern, so dass man pro Erwähnung immer wieder die jeweils aktuelle Unterstellung suchen müsste. Die Hierarchie kann bei einem Ereignis ein Schnappschuss sein, ist aber normalerweise ein Zustand über einen längeren Zeitabschnitt. Weiterführende Informationen zur Geschichte eines Ortes sind damit aber nicht speicherbar.

http://www.beholdgenealogy.com/blog/?p=899

Eine Sammlung von Orten, um deren kirchlichen, verwaltungsrechtlichen usw. Hierarchien darzustellen, finden wir auf der Internet-Seite:

http://gov.genealogy.net

GEDCOM 5.5.5 erlaubt und schlägt dazu einen benutzerdefinierten Haupt-Datensatz (Tabelle) vor, der von verschiedenen Programmen mit `_PLAC` oder `_LOC` oder anderen benutzerdefinierten Bezeichnern schon realisiert wird:

http://wiki-en.genealogy.net/Gedcom_5.5EL

2.2.2 Virtuelle Elemente für die Beziehungen

Wie werden nun die Beziehungen zwischen den physischen Elementen hergestellt? Dazu stellen wir uns gedanklich auf einen Repräsentanten (Datensatz) der jeweiligen Tabelle und formulieren pro Beziehung zwei immer für alle Repräsentanten und alle Zeiten wahre Aussagen. Dabei benutzen wir folgende Begriffe:

hat | hat genau	1:1	
hat möglicherweise einen	0:1	keinen oder einen
hat mehrere	1:m	mindestens einen
hat möglicherweise mehrere	0:m	keinen, einen oder mehrere
kann haben	0:x	es ist auch „keiner“ möglich

GEDCOM führt den Begriff „**Familie**“ (family) zur Beschreibung der Beziehungen zwischen Individuen (Personen) ein, auf die wir uns vorerst konzentrieren wollen, da alle anderen Tabellen der Ergänzung dienen.

Eine Person gehört möglicherweise zu mehreren Familie	Findelkind, Leichenfunde Kind gehört mindestens zur biologischen Elternfamilie Person kann rechtlich adoptiert werden
Eine Familie umfasst mehrere Personen	Vater, Mutter, Kind(er)
Eine Familie darf höchstens einen Vater `HUSB` und eine Mutter `WIFE` besitzen.	Es ist jedoch nicht vorgeschrieben, dass mindestens einer von beiden Personen existieren muss.
Eine Familie kann mehrere Kinder besitzen.	Also auch kein Kind.

GEDCOM bezeichnet das Schema, mit der die Beziehungen zwischen den physischen Elementen hergestellt wird als `Lineage-Linked Form` (abstammungsverknüpfte Form).

2.2.3 GEDCOM 5.5.5

GEDCOM 5.5.5 ist eine Weiterentwicklung von GEDCOM 5.5.1 AE (annotation edition), bei der der Niederländer Tamura Jones seine Erfahrungen mit der Version 5.5.1 und deren Fehler niedergelegt hat. Er sammelte und entschied ohne eigenes kommerzielles Interesse die Vorschläge der unterschiedlichen Entwickler.

Die Version 5.5.5 vom 02.10.2019 ist eine „Wartungsfreigabe. Qualität. Einfacher und strikter‘ (Maintenance release. Quality. Simpler & Stricter). Ja, das Englisch entdeckt die Vorteile der Groß-/Kleinschreibung.

Previous GEDCOM releases were feature releases that introduced major new features. GEDCOM 5.5.5 does not introduce any major new features, quite the opposite: GEDCOM 5.5.5 removes long obsolete and deprecated features. GEDCOM 5.5.5 is a maintenance release; it fixes a variety of issues to provide a better specification, and a solid basis for future versions.

Einige der Präzisierungen werden in diesem Text erwähnt. Ein Großteil der Präzisierungen dienen dazu, GEDCOM als formale Sprache genauer zu beschreiben.

Neu bei GEDCOM 5.5.5 ist die Betrachtung der Fortsetzungszeilen `CONC` und `CONT`, die eng mit dem Problem der Zeichensätze und deren Verschlüsselung verbunden sind.

Eine Reihe von Elementen ist in der Version 5.5.5 nicht mehr erlaubt.

GEDCOM 5.5.5-Schreiber sollen diese Elemente nicht mehr exportieren. GEDCOM-Leser sollten diese Elemente ignorieren oder gar die gesamte Datei als nicht 5.5.5-konform ablehnen.

Eine wichtige **Forderung** von GEDCOM 5.5.5 besteht darin, nur Dateien, die alle Vorgaben des Standards erfüllen, zu verarbeiten. Alle Abweichungen sollen verworfen werden. Jedoch scheint eine Zertifizierung von Programmen (wie bei Java) nicht vorgesehen zu sein.

Eine wesentliche **Änderung** gegenüber der alten Philosophie sollte unbedingt erwähnt werden:

The text of the specification no longer assumes that recorded relationships are biological. Relationships recorded on the basis of official paper documentation are official, and should not be assumed to be biological.

Ich halte diese Aussage für einen wesentlichen Paradigmenwechsel für die Zukunft, weg von der idealen Familie mit biologischer Vererbung hin zu den realen Gegebenheiten mit juristischen Vererbung.

GEDCOM 5.5.5 sieht einige wichtige **Vereinfachungen** und **Vereinheitlichungen** insbesondere für Eigenschaften vor, die eher selten eingesetzt werden bzw. weitgehend selbstverständlich sind:

- Einheitlicher Inhalt, keine Mischung von Versionen, keine unterschiedliche Zeilenschaltungen, keine unterschiedlichen Verschlüsselungen, keine unterschiedlichen Zeichensätze pro Datenbank
- GEDCOM Tags stehen immer in Großbuchstaben
- usw.

GEDCOM 5.5.5 entrümpelt die Tags, die wenig gebräuchlich, redundant oder widersprüchlich sind.

GEDCOM 5.5.5 führt einige neue Elemente bzw. erlaubte Werte ein, beispielsweise:

Das Geschlecht `SEX` kann jetzt auch `X` (divers) oder `N` (unbekannt) sein. Geburtstage müssen kein Jahr enthalten (wenn man nur gefeiert hat). Partnerschaften können allgemein und registriert sein.

Eine Vielzahl von Entwurfsfehlern wird ab S. 66 aufgeführt und korrigiert. Die detaillierte Darstellung würde den Rahmen dieses Skripts sprengen:

Wohnen (`RESI`) ist kein Ereignis. Es beschreibt einen zeitlich länger dauernden Zustand, die mit dem Ereignis *Einzug* beginnt und mit dem *Auszug* endet.

Alter (`AGE_AT_EVENT`) ist für Ereignisse aber nicht für Perioden (`RESI, OCCU`) gültig. Im Grunde sind alle Alters-Attribute in GEDCOM berechenbare Werte, die aus Datumsangaben abgeleitet werden könnten. Da aber letztere weder exakt noch geschätzt angegeben werden, lässt GEDCOM an vielen Stellen dieses Attribut zu.

Die Existenz von Rufnamen wird in GEDCOM 5.5.5 explizit angesprochen. Dabei wird auf `_RUFNAME` verwiesen.

2.2.4 GEDCOM EL

Um einige der gröbsten Unzulänglichkeiten des GEDCOM-Standards zu beseitigen, hat der *Verein für Computergenealogie e. V.* den Vorschlag GEDCOM 5.5 EL (Extended Location) erarbeitet, der

- auf der letzten verabschiedeten Version aufsetzt
- und hauptsächlich die fehlende Ortstabelle beinhaltet.

Zwar steht der Vorschlag jedem zur Verfügung, scheint aber nur bei den deutschen Progammentwicklern auf offene Ohren gestoßen zu sein.

2.3 Grammatik zur Darstellung der Daten

2.3.1 Konzept

Eine GEDCOM Übertragung (transmission) besteht aus einer Sequenz logischer Datensätze, wovon jeder wiederum aus einer Sequenz von GEDCOM-Zeilen besteht, die wiederum alle in einer sequenziellen Datei oder Datenstrom von Zeichen enthalten sind.

Die Verwendung des Begriffs *transmission* in GEDCOM 5.5.1 unterstreicht die falsche Annahme, dass die Informationen nicht unbedingt als Dateien sondern auch in einem kontinuierlichen Datenstrom übertragen werden können. Diese Übersetzung aus GEDCOM 5.5.1 ist zweideutig und könnte fälschlicherweise als Übertragungsprotokoll aufgefasst werden (vergleiche `http` als Protokoll und `html` als Beschreibungssprache).

Im Gegensatz zu einer Datenbanktabelle mit fest vorgegebenen Feldbreiten werden „logische Datensätze“ übertragen, die wiederum aus für den Menschen lesbaren Zeilen bestehen und verschieden lang sind. Im Unterschied zu einem freien Text gibt es eine vorgegebene (halbformale) Struktur mit Schlüsselwörtern (tags).

Die Tags sind nicht global eindeutig bezeichnet. Vielmehr hängt die Bedeutung eines Tags auch davon ab, in welchem Kontext er steht. Um einen Tag eindeutig zu machen, wird er **qualifiziert**, d. h., mit dem übergeordneten Tag versehen. `HEAD.SUBM` ist der Einreicher im Kopf, d. h., der Erzeuger (das Erzeugerprogramm) der GEDCOM-Datei. `INDI.SUBM` ist der Verfasser-Datensatz, der die Daten zu einem Individuum zusammengetragen hat.
Die Qualifizierung kann auch mehrere Stufen (Ebenen) umfassen `HEAD.SOUR.DATA`, wenn das Element weitere Unterelemente enthalten darf.

Die Grundstruktur einer GEDCOM-Datei besteht aus *Haupt-Datensätzen* (top-level records), die jeweils aus einem Block von Zeilen (Unterdatensätze, subrecords) bestehen:

```
0 <<HEADER>>                {1:1}
0 <<SUBMISSION_RECORD>>     {0:1}
0 <<RECORD>>                {1:M}
0 TRLR                      {1:1}
```

Somit besteht eine Datei aus genau einer Kopfstruktur (`HEADER`), einem optionalen Einreichungs-Datensatz, mindestens einem (Informations-)Datensatz (`RECORD`) und genau einer Abschlusszeile (`TRAILER`). Die Strukturen sind Nichterminale (weiter zerlegbar), `TRLR` ist ein Terminalsymbol, alle anderen Elemente werden weiter verfeinert (Nichtterminalsymbole). In der rechten Spalte werden die Multiplizitäten (UML-Bezeichnung) oder die wesensgleichen Kardinalitäten (ERM-Bezeichnung) angegeben.

`TRLR` wurde von GEDCOM 5.5.1 dazu benutzt, um den Stammbaum auf mehrere getrennte GEDCOM-Dateien verteilen zu können. GEDCOM 5.5.5 erlaubt dies nicht und verlangt das `BOM` als erstes Zeichen, wodurch `TRLR` ist eigentlich überflüssig wird. Es hilft jetzt aber, unvollständige GEDCOM-Dateien zu erkennen.

2.3.2 Grammatikregeln

Die Maßeinheit der GEDCOM-Spezifikation war das **Zeichen** (character). Das ist aber keine absolute Maßeinheit. Ein Zeichen kann je nach Verschlüsselung mit bis zu vier Bytes gespeichert werden. GEDCOM 5.5.5 führt dazu die *Kodeeinheit* (code unit) ein.

Ein *logischer Datensatz* (top-level record, Block mit mehrerer Zeilen (records)) beginnt mit der Ebenennummer 0 (Ziffer Null). Er darf maximal 32 K Zeichen lang sein.

Hier gibt es schon ein Problem: Was ist ein Zeichen? Ist ein Ä ein Zeichen oder ein aus zwei Zeichen zusammengesetzte Glyphe? Programmierer legen (Eingangs-)Puffer in Bytes an. Die Datenübertragung erfolgt in „logisch" kontinuierlichem Datenstrom (auch wenn dann daraus Pakete gemacht werden). Diese Beschränkung stammt wohl noch aus dem „DOS-Zeitalter" mit 16 Bit-Adressen, als ein 32 KB-Block die Hälfte des Adressraums umfasste. Der GEDCOM-Standard wurde nicht von einem Experten geschrieben.

Gleichzeitig schränkt diese Obergrenze die Möglichkeiten ein, große Binärdateien in einer GEDCOM-Datei einzubetten (siehe dazu die Unterschiede von Version 5.5 und 5.5.1 bei der Multimediastruktur an), auch wenn wir verzweifelt versuchen, diese in 32 KB-Blöcke zu zerlegen und zu verknüpfen.

Alle weiteren Zeilen, die zum Haupt-Datensatz (top-level record) gehören, haben eine höhere Ebenennummer (1 bis 99, Ziffern ohne führende Nullen). Dabei dürfen keine Lücken entstehen, d. h., die Ebenennummern steigen jeweils um 1.

Jeder Datensatz enthält entweder einen Wert oder stellt den Beginn eines Unter-Datensatzes (mit Folgenummern) dar.

Die Verknüpfung der Datensätze untereinander erfolgt über IDs, die einen Datensatz eindeutig benennen (Referenz-IDs) sowie über Verweis-IDs (Zeiger), die auf diese IDs zeigen. Diese IDs sind mit At-Zeichen (`@`) umklammert. Referenz-IDs tauchen nur in Haupt-Datensätzen (top-level records) auf. Die ID kann alphanumerisch sein. GEDCOM 5.5.1 erlaubte sogar Sonderzeichen und beschreibt die Verwendung des Ausrufezeichens. Es hat sich durchgesetzt, die IDs mit einem Anfangsbuchstaben zu klassifizieren, also `@I123@` für ein Individuum, `@F123@` für eine Familie usw.

Ein Zeiger ist ***legal***, wenn er die Syntax erfüllt. Ein Zeiger ist ***gültig***, wenn auf die Referenz-ID eines Datensatzes vom richtigen Typ verweist. Die Richtung des Verweises ist dabei ohne Bedeutung.

Die Länge der GEDCOM-Zeile darf maximal 255 Zeichen betragen. Daher muss es einen Mechanismus geben, lange Zeilen aufzuspalten. Der Empfänger benötigt somit (nur) einen Eingangspuffer dieser Länge.

Hier gibt es wieder das Problem: Aufgrund der „modernen" Verschlüsselung von Texten in Unicode, UTF-8, UTF-16 usw. kann der Speicherbedarf aufgrund der Zeichen nicht vorausberechnet werden. Erst nach der Verschlüsselung ist die tatsächliche Länge bekannt. Der Speicherbedarf liegt dann irgendwo zwischen der Zeichenzahl (ASCII) und der doppelten Zeichenzahl (UTF-16). Selbst im proprietären GEDCOM-ANSEL ist die Länge nicht a priori wegen der jeweils vorangestellten diakritischen Zeichen voraussagbar.

Da die Zeilen von Programmen erstellt werden sollen, gibt es keine Kommentare, eine Einrückungen (Leerräume) und auch keine Leerzeilen.

Die GEDCOM-Datei **stellungsorientiert**, d. h., die Stellung der einzelnen Zeilen zueinander enthält Information.

Hier folgt das nächste Problem, da diese Behauptung nicht ganz zutrifft. Der Tag `Name` ist auch inhaltsorientiert, da er die Vornamen und den Nachnamen in Schrägstrichen einschließt. Somit muss dieser Tag noch einmal gesondert behandelt werden, indem Zeichen für Zeichen untersucht werden muss. Einige Ahnenprogramme zeigen daher fehlende Namen mit zwei aufeinanderfolgenden Schrägstrichen an.

Beispiel:

Eine große GEDCOM-Datei enthielt „verdrehte“ `NAME`-Zeilen, einmal in „richtiger“ Reihenfolge `Vorname /Nachname/`, einmal in falscher Reihenfolge `/Nachname/ Vorname`. Es gibt Programme, bei denen man die Reihenfolge beim Export bzw. Import einstellen kann. Mischt man nun zwei unterschiedlich exportierte GEDCOM-Dateien, so orientiert sich der Mischer erst einmal nur am `INDI`-Tag und nicht am Inhalt. Der Auftrag war klar: Reihenfolge vereinheitlichen.

Um den Datenumfang (für den Austausch der Datei im Internet) zu reduzieren, wurden die verdrehten Daten eingelesen und mit Hilfe der Exportfunktion wieder ausgegeben. Ein besonderes Feature besteht bei vielen Programmen darin, dass der Benutzer seine Exporte gezielt einschränken kann (wegen des Datenschutzes?). Also wurden nur die allernotwendigsten Daten natürlich mi den falschen Namen exportiert.

Aber was passiert dann? Die reparierte GEDCOM-Datei enthält zwar die reparierten Namen, aber alle anderen Informationen sind nicht mehr zuordenbar. Beim Einlesen der reparierten Datei fehlen alle Anmerkungen, alle Orte usw., je nachdem was nicht exportiert wurde.

Die GEDCOM-Grammatik ist **kontextsensitiv**, d. h., dass einzelne Nichtterminalsymbole nur in einem vorgegebenen Kontext (logischen Datensatz) ersetzt werden dürfen.

2.3.3 Grammatiksyntax

Eine GEDCOM-Zeile hat folgende Syntax:

```
Gedcom_Zeile := Ebene + [Begr + Referenz_ID] + Begr + Kennzeichen + [Begr + Zeilenwert] + Zeilenende
```

Wir lesen:

:=	besteht aus, wird ersetzt durch
+	gefolgt von
|	oder
[]	optionale Elemente

Beispiel:

```
1 NAME Hans /Huber/
   2  GIVN Peter
   2  SURN Mayer
```

Und schon haben wir ein Beispiel, das gegen die 1NF (1. Normalform) der Datenbanktheorie verstößt. Das Feld `NAME` ist nicht atomar. Es besteht aus mindestens zwei Elementen, wobei noch eines (der Nachname) besonders mit Schrägstrichen im Text gekennzeichnet ist.

Aber auch „Zeilenende“ (terminator) ist nicht trivial, so dass wir hier ein gesondertes Kapitel einschieben müssen.

2.3.4 Redundanz und Widerspruch (referenzielle Integrität)

Bei der Betrachtung der vielen Tags stellen wir sehr schnell fest, dass in der Version 5.5.1 eine Information an verschiedenen Stellen festgelegt sein kann. Nutzen wir alle Möglichkeiten, dann erzeugen wir damit deutliche Redundanzen.

Umgekehrt besteht bei Mehrfachspeicherung die Gefahr von Widersprüchen. Dies ist im letzten Beispiel eines Personensatzes zu sehen. Welche der beiden Informationen über Vor- und Zuname ist denn nun richtig?

Existiert zu jedem Verweis auch der passenden Datensatz?

Stimmt die Familienzugehörigkeit einer Person mit der Personenliste in der Familie überein?

GEDCOM 5.5.5 ist der Versuch, die Auswahlmöglichkeiten auf jeweils nur eine zu reduzieren.

2.3.5 Problem: Huhn oder Ei?

Bei den durch Verweis-IDs verknüpften Datensätzen ergibt sich das Problem der **referenziellen Integrität**. Die Angabe einer Verweis-ID setzt voraus, dass das Zielobjekt mit der gleichen Referenz-ID auch wirklich vorhanden ist. GEDCOM 5.5.5 sagt: die Verweis-ID muss „gültig“ sein.

Beim Einlesen einer GEDCOM-Datei in einem Ein-Pass-Durchlauf müssten also alle Zielobjekte angelegt sein, bevor wir aus sie verweisen können, wenn wir gleichzeitig referenzielle Integrität verlangen.

Beispiel: Besitzt eine Person eine `FAMC`-Zeile (ist Kind in Familie) mit einer Verweis-ID, so muss der `FAM`-Datensatz mit derselben Referenz-ID bereits vorher angelegt sein.

Nun darf aber der `FAM`-Datensatz Verweise auf Personen in den Rollen `HUSB, WIFE, CHIL` enthalten, die wiederum noch nicht angelegt sind und in der GEDCOM-Datei möglicherweise erst später gespeichert sind.

Ein Ein-Pass-Verfahren scheidet also aus. Das Problem kann ein Programm durch zwei Techniken lösen:

1. Die GEDCOM-Datei wird nur einmal durchlaufen. Die IDs werden wie vorgegeben ohne Prüfung in eine Nachschautabelle eingetragen. Die referenzielle Integrität wird nachträglich geprüft und ggf. mit Fehlermeldungen angemahnt (reaktiv).
2. Die GEDCOM-Datei wird zweimal durchlaufen (Zwei-Pass-Verfahren). Beim ersten Durchlauf werden verschiebliche (relocatable) IDs eingetragen, die beim zweiten Durchlauf durch die echten IDs ersetzt werden (präventiv).

Das Einlesen wird noch einmal schwerer, wenn man schon eine Datenbank hat und die neuen GEDCOM-Daten einmischen will.

2.4 Verwaltungsdaten

Zur Verarbeitung der genealogischen Daten sieht das GEDCOM-Format Beschreibungsdaten (Meta-Daten) vor.

Die Gesamtstruktur einer GEDCOM-Datei sieht folgendermaßen aus:

```
0 <<HEADER>>                {1:1}
0 <<SUBMISSION_RECORD>>     {0:1}
0 <<RECORD>>                {1:M}
0 TRLR                      {1:1}
```

Sie muss mit einem (komplexen) Kopf `<<HEADER>>` beginnen. Beschreibt mit `<<SUBMISSION_RECORD>>` optional die Einreichung. Enthält mindestens einen Datensatz `<<RECORD>>`. Schließt mit einer Abschlusszeile ab.

2.4.1 Kopf

Die GEDCOM-Datei beginnt mit einem obligaten Kopf:

```
0 <<HEADER>>                {1:1}
0 <<SUBMISSION_RECORD>>     {0:1}
0 <<RECORD>>                {1:M}
0 TRLR                      {1:1}
```

Der Kopf muss also nach Standard vorhanden sein:

```
HEADER :=
  n HEAD                                          {1:1}
    +1 SOUR <APPROVED_SYSTEM_ID>                  {1:1}
      +2 VERS <VERSION_NUMBER>                    {0:1}
      +2 NAME <NAME_OF_PRODUCT>                   {0:1}
      +2 CORP <NAME_OF_BUSINESS>                  {0:1}
        +3 <<ADDRESS_STRUCTURE>>                  {0:1}
      +2 DATA <NAME_OF_SOURCE_DATA>               {0:1}
        +3 DATE <PUBLICATION_DATE>                {0:1}
        +3 COPR <COPYRIGHT_SOURCE_DATA>           {0:1}
    +1 DEST <RECEIVING_SYSTEM_NAME>               {0:1*}
    +1 DATE <TRANSMISSION_DATE>                   {0:1}
      +2 TIME <TIME_VALUE>                        {0:1}
```

```
+1 SUBM @<XREF:SUBM>@                                  {1:1}
+1 SUBN @<XREF:SUBN>@                                  {0:1}
+1 FILE <FILE_NAME>                                    {0:1}
+1 COPR <COPYRIGHT_GEDCOM_FILE>                        {0:1}
+1 GEDC                                                {1:1}
  +2 VERS <VERSION_NUMBER>                             {1:1}
  +2 FORM <GEDCOM_FORM>                                {1:1}
+1 CHAR <CHARACTER_SET>                                 {1:1}
  +2 VERS <VERSION_NUMBER>                             {0:1}
+1 LANG <LANGUAGE_OF_TEXT>                             {0:1}
+1 PLAC                                                {0:1}
  +2 FORM <PLACE_HIERARCHY>                            {1:1}
+1 NOTE <GEDCOM_CONTENT_DESCRIPTION>                   {0:1}
  +2 [CONT|CONC] <GEDCOM_CONTENT_DESCRIPTION>          {0:M}
```

** NOTE:*
Submissions to the Family History Department for Ancestral File submission or for clearing temple ordinances must use a `DEST`ination of `ANSTFILE` or TempleReady.

The header structure provides information about the entire transmission. The `SOUR`ce system name identifies which system sent the data. The `DEST`ination system name identifies the intended receiving system.

Die Einrückungen dienen nur der besseren Lesbarkeit und treten in GEDCOM-Dateien nicht auf.

Die Anmerkung zeigt den wesentlichen Zweck einer GEDCOM-Datei auf. Der Kopf enthält keine Strukturen aber zwei Verweise `SUBM` und `SUBN`, auf die Einreicher und mehrere atomare Elemente.

`GEDC` beschreibt die Version und das Schema (`FORM`) der Datei, wobei aktuell nur der Wert `Lineage-Linked` erlaubt ist. Dieser Datensatz scheint also mehrere verschiedene Schemata zuzulassen, was jeden Programmierer unruhig werden lässt.

`PLAC` legt Informationen für die Ortsangabe fest. `PLAC.FORM` beschreibt dabei, in welcher Reihenfolge die juristische Hierarchie angegeben wird. Beispiel: `2 FORM Stadt, Kreis, Bezirk, Land, Staat`

2.4.2 Einreichung

Das sendende System nutzt einen „Einreichungs-Datensatz“, um dem empfangenden System Instruktionen und Informationen zu übermitteln. `TempleReady` verarbeitet diese, um zu bestimmen, zu welchem Tempel die freigegebenen Datensätze weitergeleitet werden. Der `Submissions`-Datensatz wird auch für die Kommunikation zwischen Ancestral File und `TempleReady` genutzt. Jede GEDCOM Übertragung muss genau einen `Submissions`-Datensatz enthalten. Mehrere Einreichungen werden in separate GEDCOM-Übermittlungen aufgetrennt.

Die Einreichung dient hauptsächlich zur Übermittlung von Datensätzen an Einrichtungen des LDS über das Programm `TempleReady`. Für den privaten Datenaustausch ist dieser Datensatz nicht unbedingt nötig.

2.4.3 Abschluss

Den Abschluss einer GEDCOM-Datei bildet eine Zeile der Form

```
0 TRLR
```

mit dem Terminalsymbol `TRLR`.

2.5 Tabellen

In diesem Kapitel wollen wir die einzelnen (wichtigen) Tabellen näher betrachten und einer kritischen Prüfung unterziehen. Dabei werden wir feststellen, dass es eine Vielzahl von komplexen Konstrukten gibt, die so gut wie von keinem Programm genutzt wird.

Die GEDCOM-Datei besteht aus *Datensätzen* (records), die jeweils einen Block von Zeilen bilden. Blöcke können geschachtelt werden. Mit jedem Block erhöht sich die Ebenennummer um 1. Am Blockende springt die Ebenennummer ggf. um einen größeren auf einen kleineren Wert zurück.

2.5.1 Person (individual)

Für eine Person sieht der GEDCOM 5.5.1 Standard einen Datensatz mit folgender Struktur vor:

```
INDIVIDUAL_RECORD :=
  n @XREF:INDI@ INDI                                {1:1}
    +1 RESN <RESTRICTION_NOTICE>                    {0:1}
    +1 <<PERSONAL_NAME_STRUCTURE>>                  {0:M}
    +1 SEX <<SEX_VALUE>>                            {0:1}
    +1 <<INDIVIDUAL_EVENT_STRUCTURE>>               {0:M}
    +1 <<INDIVIDUAL_ATTRIBUTE_STRUCTURE>>           {0:M}
    +1 <<LDS_INDIVIDUAL_ORDINANCE>>                 {0:M}
    +1 <<CHILD_TO_FAMILY_LINK>>                     {0:M}
    +1 <<SPOUSE_TO_FAMILY_LINK>>                    {0:M}
    +1 SUBM @<XREF:SUBM>@                           {0:M}
    +1 <<ASSOCIATION_STRUCTURE>>                    {0:M}
    +1 ALIA @<XREF:INDI>@                           {0:M}
    +1 ANCI @<XREF:SUBM>@                           {0:M}
    +1 DESI @<XREF:SUBM>@                           {0:M}
    +1 RFN <PERMANENT_RECORD_FILE_NUMBER>           {0:1}
    +1 AFN <ANCESTRAL_FILE_NUMBER>                  {0:1}
    +1 REFN <USER_REFERENCE_NUMBER>                 {0:M}
    +2 TYPE <USER_REFERENCE_TYPE>                   {0:1}
    +1 RIN <AUTOMATED_RECORD_ID>                    {0:1}
    +1 <<CHANGE_DATE>>                              {0:1}
    +1 <<NOTE_STRUCTURE>>                           {0:M}
    +1 <<SOURCE_CITATION>>                          {0:M}
    +1 <<MULTIMEDIA_LINK>>                          {0:M}
```

Bis auf die Tag-Zeile **INDI** sind alle Elemente optional. Der Standard ergänzt folgende Regeln (Auszug):

Der Personendatensatz (INDIVIDUAL_RECORD) ist eine Sammlung von bekannten oder ermittelten Fakten zu einer Person. Manchmal stammen die Fakten aus unterschiedlichen Quellen. Diese Form erlaubt die Dokumentation der jeweiligen Quelle, in der ein Fakt entdeckt wurde.

Die normalen Abstammungsverbindungen werden durch Zeiger von der Person auf eine Familie durch entweder einen FAMC-Kennzeichen oder FAMS-Kennzeichen gezeigt. Das FAMC-Kennzeichen beinhaltet einen Zeiger auf eine Familie, in der die Person Kind ist. Das FAMS-Kennzeichen beinhaltet einen Zeiger auf eine Familie, in der die Person (Ehe-)Partner oder Elternteil ist. Die <<CHILD_TO_FAMILY_LINK>> Struktur enthält einen FAMC-Zeiger, der zwingend notwendig ist, um die Verbindung eines Kindes zu seinen Eltern zu dokumentieren. Die <<CHILD_TO_FAMILY_LINK>> Struktur zeigt zudem, ob es sich um eine biologische Verbindung, Adoption oder Siegelung handelt.

Die CHILD_TO_FAMILY_LINK-Struktur ist in erster Linie ein FAMC-Verweis auf eine Familie mit dem optionalen Zusatz der Rolle des Kindes in der Familie. Eine Person kann zu mehreren Familien gehören

```
CHILD_TO_FAMILY_LINK :=
    n FAMC @<XREF:FAM>@                             {1:1}
     +1 PEDI <PEDIGREE_LINKAGE_TYPE>                {0:1}
     +1 STAT <CHILD_LINKAGE_STATUS>                 {0:1}
     +1 <<NOTE_STRUCTURE>>                          {0:M}
```

Analog enthält die <<SPOUSE_TO_FAMILY_LINK>>-Struktur einen FAMS-Verweis auf die Familie für die Rolle eines Elternteils.

```
SPOUSE_TO_FAMILY_LINK :=
    n FAMS @<XREF:FAM>@                             {1:1}
    +1 <<NOTE_STRUCTURE>>                           {0:M}
```

Die Vielzahl der Mehrfachfelder mit der Multiplizität {0:M} springt natürlich sofort ins Auge. So kann eine Person beliebig viele <<PERSONAL_NAME_STRUCTURE>>-Strukturen besitzen, die den Namen einer Person festlegen.

Dazu werden zuerst Namensstücke deklariert, die erneut Strukturen zu den Anmerkungen und Quellen enthalten können:

```
PERSONAL_NAME_PIECES :=
   n NPFX <NAME_PIECE_PREFIX>                          {0:1}
   n GIVN <NAME_PIECE_GIVEN>                           {0:1}
   n NICK <NAME_PIECE_NICKNAME>                        {0:1}
   n SPFX <NAME_PIECE_SURNAME_PREFIX>                  {0:1}
   n SURN <NAME_PIECE_SURNAME>                         {0:1}
   n NSFX <NAME_PIECE_SUFFIX>                          {0:1}
   n <<NOTE_STRUCTURE>>                                {0:M}
   n <<SOURCE_CITATION>>                               {0:M}
```

Diese Unterunterstruktur dann in der Struktur verwandt werden (die optischen Einrückungen sind in GEDCOM nicht vorgesehen):

```
PERSONAL_NAME_STRUCTURE :=
n NAME <NAME_PERSONAL>                                 {1:1}
   +1 TYPE <NAME_TYPE>                                 {0:1}
   +1 <<PERSONAL_NAME_PIECES>>                         {0:1}
   +1 FONE <NAME_PHONETIC_VARIATION>                   {0:M}
      +2 TYPE <PHONETIC_TYPE>                          {1:1}
      +2 <<PERSONAL_NAME_PIECES>>                      {0:1}
   +1 ROMN <NAME_ROMANIZED_VARIATION>                  {0:M}
      +2 TYPE <ROMANIZED_TYPE>                         {1:1}
      +2 <<PERSONAL_NAME_PIECES>>                      {0:1}
```

Der Standard schreibt weiter vor:

Der Name wird so gebildet, wie er normalerweise ausgesprochen wird, wobei die Vornamen und der Familienname durch Schrägstriche (/) getrennt werden. (Siehe <NAME_PERSONAL>) Weil Namen veränderlich oder auf unbekannte Weise zusammengesetzt werden können, ist es schwierig, eine detailliertere Struktur für Namensteile bereitzustellen, die jeden Fall abdeckt. Die Kennzeichen NPFX, GIVN, NICK, SPFX, SURN und NSFX werden für solche Systeme bereitgestellt, die nicht effektiv mit weniger strukturierten Informationen umgehen können. Um die weitere Kompatibilität für die Zukunft zu gewährleisten, müssen alle Systeme den Namen aus der <NAME_PERSONAL> Struktur zusammensetzen. Programme, welche die optionalen Namensfelder nutzen, sollten davon ausgehen, dass nur wenige Systeme diese verarbeiten, und die meisten diese nicht selbst bereitstellen.

<NAME_TYPE> wird dafür benutzt, um anzugeben, um welche Art von Namensvariation es sich handelt. Wenn <NAME_TYPE> beispielsweise unterhalb von <NAME_PERSONAL> ist, so könnte er anzeigen, dass die Person den Namen bei der Immigration angenommen hat, oder dass es sich um einen Alias handelt

Hier schränkt sich der Standard schon einmal selbst mit Annahmen ein, so dass sich zwei unterschiedliche Programme mit hoher Wahrscheinlichkeit nicht verstehen werden.

Nach der Beschreibung ist der <NAME_PERSONAL> ein zusammengesetztes (nicht atomares) Element, das somit gegen die 1NF des Relationalen Datenmodells verstößt. Es enthält das Trennzeichen /, das wir daher nicht mehr für die oft verwendete Abtrennung unterschiedlicher Schreibweisen benutzen können. Aliasnamen müssen also getrennt angegeben werden. Hierzu stehen uns jedoch die Variationen offen.

Interessant ist beispielsweise, dass der Standard auch romanisierte Namensteile vorsieht, die auch noch mehrfach vorkommen dürfen und die auf jeder Tastatur erzeugbar sein sollten. Damit können Aliasnamen erfasst werden, bei denen einfach die diakritischen Elemente entfernt wurden, aber auch solche, bei denen die Aussprache transformiert wurde:

Beispiel: Lička, Licka, Liska, Lischka

2.5.2 Familie (family)

Für eine Familie sieht der GEDCOM Standard einen Datensatz mit folgender Struktur vor:

```
FAM_RECORD :=
   n @<XREF:FAM>@ FAM                                  {1:1}
    +1 RESN <RESTRICTION_NOTICE>                       {0:1}
    +1 <<FAMILY_EVENT_STRUCTURE>>                      {0:M}
    +1 HUSB @<XREF:INDI>@                              {0:1}
    +1 WIFE @<XREF:INDI>@                              {0:1}
    +1 CHIL @<XREF:INDI>@                              {0:M}
    +1 NCHI <COUNT_OF_CHILDREN>10                      {0:1}
```

```
+1 SUBM @<XREF:SUBM>@                          {0:M}
+1 <<LDS_SPOUSE_SEALING>>                      {0:M}
+1 REFN <USER_REFERENCE_NUMBER>                {0:M}
 +2 TYPE <USER_REFERENCE_TYPE>                 {0:1}
+1 RIN <AUTOMATED_RECORD_ID>                   {0:1}
+1 <<CHANGE_DATE>>                             {0:1}
+1 <<NOTE_STRUCTURE>>                          {0:M}
+1 <<SOURCE_CITATION>>                         {0:M}
+1 <<MULTIMEDIA_LINK>>                         {0:M}
```

Auch dieser Datensatz kann bis auf die Tag-Zeile entsprechend der Multiplizitäten leer sein.

Der Standard schreibt weiter vor:

Der Familien-Datensatz wird dazu benutzt, kirchliche und standesamtliche Ehen aufzuzeichnen, sowie Paare die durch gemeinsame Elternschaft entstehen. Es darf nicht mehr als ein Ehemann/Vater (`HUSB`) und eine Ehefrau/Mutter (`WIFE`) in einem `FAM_RECORD` erscheinen.

Wenn beispielsweise ein Mann mehrmals verheiratet war, so würde er in mehreren `FAM_RECORDs` erscheinen. Die `FAM_RECORD`-Struktur nimmt an, dass der Ehemann (`HUSB`) männlich ist, und die Ehefrau (`WIFE`) weiblich.

Die bevorzugte Reihenfolge der Kinder-Zeiger (`CHIL`) innerhalb des Familien-Datensatzes ist die chronologische nach Geburtsdatum der Kinder.

Auch hier soll daran erinnert werden, dass das Kind einen abweichenden oder fehlenden `FAMC`-Zeiger haben kann. Einige Programme verzichten daher ganz auf einen der beiden Zeiger. Im RDM stellen die `CHIL`-Zeiger ein unzulässiges Wiederholfeld dar, so dass nur der `FAMC`-Zeiger bei den Kindern erlaubt ist.

Wichtig für den praktischen Einsatz ist die `<<FAMILY_EVENT_STRUCTURE>>`, in der wir das Heiratsdatum und ggf. das Alter der Eheleute zum Heiratsereignis eintragen können. Bei letzteren Einträgen stellt sich wieder die Frage nach

- Redundanz: Das Alter ist aus dem Geburtstag und dem Heiratsdatum berechenbar.
- Widerspruch: Das angegebene Alter und das berechnete können sich widersprechen.
- Unschärfe: Altersangabe, Geburtsdatum, Heiratsdatum sind möglicherweise nur unscharf bestimmt.

Aber es lassen sich noch weitere berechenbaren Informationen finden. So lässt sic h die Anzahl der Kinder `NCHI` durch Mitzählen beim Einlesen der GEDCOM-Datei ermitteln. Eine abweichende Zahl soll die Möglichkeit öffnen, auf noch unbekannte Kinder aufmerksam zu machen.

Die `<<FAMILY_EVENT_STRUCTURE>>` darf mehrfach erscheinen. Damit ist es möglich, kirchliche und standesamtliche Trauung sowie das Ende der Familie abzuspeichern. Dabei stellt sich die Frage, ob eine Ehe nach ihrem Ende wieder aufleben kann. Damit dürfen die Ereignisse erneut in zeitlich richtiger Abfolge erneut erscheinen, ansonsten wären Doppelungen fehlerhaft.

2.5.3 Medien (multimedia)

Die Tabelle mit den Zugriffen auf die Medien hat mit der Version 5.1.1 die größten Änderungen erlebt.

Für ein Medium sieht der GEDCOM-Standard 5.5 einen Datensatz mit folgender Struktur vor:

Version 5.5

```
MULTIMEDIA_RECORD :=
n @<XREF:OBJE>@ OBJE                           {1:1}
  +1 FORM <MULTIMEDIA_FORMAT>                  {1:1}
  +1 TITL <DESCRIPTIVE_TITLE>                  {0:1}
  +1 <<NOTE_STRUCTURE>>                        {0:M}
  +1 <<SOURCE_CITATION>>                       {0:M}
  +1 BLOB                                      {1:1}
    +2 CONT <ENCODED_MULTIMEDIA_LINE>          {1:M}
  +1 OBJE @<XREF:OBJE>@ /*chain to continued
  object*/                                     {0:1}
  +1 REFN <USER_REFERENCE_NUMBER>              {0:M}
    +2 TYPE <USER_REFERENCE_TYPE>              {0:1}
  +1 RIN <AUTOMATED_RECORD_ID>                 {0:1}
  +1 <<CHANGE_DATE>>                           {0:1}
```

Version 5.5.1

```
MULTIMEDIA_RECORD :=
 n @XREF:OBJE@ OBJE                            {1:1}
   +1 FILE <MULTIMEDIA_FILE_REFN>              {1:M}
     +2 FORM <MULTIMEDIA_FORMAT>               {1:1}
       +3 TYPE <SOURCE_MEDIA_TYPE>             {0:1}
     +2 TITL <DESCRIPTIVE_TITLE>               {0:1}
   +1 REFN <USER_REFERENCE_NUMBER>             {0:M}
     +2 TYPE <USER_REFERENCE_TYPE>             {0:1}
   +1 RIN <AUTOMATED_RECORD_ID>                {0:1}
   +1 <<NOTE_STRUCTURE>>                       {0:M}
   +1 <<SOURCE_CITATION>>                      {0:M}
   +1 <<CHANGE_DATE>>                          {0:1}
```

Version 5.5	Version 5.5.1
Large whole multimedia objects embedded in a GEDCOM file would break some systems. For this purpose, large multimedia files should be divided into smaller multimedia records by using the subordinate OBJE tag to chain to the next <MULTIMEDIA_ RECORD> fragment. This will allow GEDCOM records to be maintained below the 32K limit for use in systems with limited resources.	*Der BLOB-Kontext wurde in 5.5.1 entfernt. Eine Referenz auf eine Multimediadatei wurde zur Struktur hinzugefügt. Es können mehrere Referenzen angegeben werden, sodass mehrere Dateien gruppiert werden können, und dabei den gleichen Kontext erhalten. Wenn man beispielsweise eine Klangdatei und ein Foto assoziieren möchte, würde man jeweils die Datei referenzieren, und jeweils dessen Format im untergeordneten FORM-Kennzeichen angeben.*

Hier weicht die Version 5.5 erheblich von der Version 5.5.1 ab. Version 5.5.5 korrigiert noch einmal und lässt pro Datensatz nur noch den Verweis auf eine Datei zu.

Offensichtlich hat man eingesehen, dass man keine BLOBs (Binary Large Object) einfach in eine reine Textdatei aufnehmen kann. Diese müssten noch einmal umkodiert werden, da eine binäre (Bild-)Datei auch alle Steuerzeichen des ASCII enthalten kann, wodurch die Datei für einen Editor nicht mehr lesbar wird. Schaut man sich eine Binärdatei (einfach einmal eine .EXE-Datei) mit einem Editor an, so reagiert dieser meist unwirsch auf `EOT` (End of Text/Transmission), `EOF` (End of File) und andere Steuerzeichen. Es ist auch schwer vorstellbar, wie Megabyte-große Bilddateien mit Fortsetzungszeilen zu je 255 Zeichen abgespeichert werden können. Das BLOB war sogar ein Mussfeld. In der Version 5.5.1 ist dagegen nur die Referenz auf eine externe Datei übrig geblieben. Somit ist ein Stammbaum nur mit der GEDCOM-Datei möglicherweise unvollständig.

Alle Medien werden verlinkt und müssen bei der Übergabe einer GEDCOM-Datei getrennt zusammengestellt und übergeben werden. Die Links bestehen aber aus absoluten Dateipfaden, die eine genaue Datenstruktur beim Empfänger voraussetzen. Schon der Wechsel zu einem anderen Laufwerk führt zu Problemen. Dies könnte nur durch (zum GEDCOM-Ordner) relativ angegebene Pfade gelöst werden.

2.5.4 Anmerkungen (notes)

Die Anmerkungen (Notizen) bestehen normalerweise aus einer Vielzahl von verketteten Zeilen, in denen der Benutzer beispielsweise Teile des Originaltextes ablegt. Anmerkungen können an vielen Stellen eingefügt werden.

Version 5.5

```
NOTE_RECORD :=
n  @<XREF:NOTE>@ NOTE <SUBMITTER_TEXT>          {1:1}
  +1 [ CONC | CONT] <SUBMITTER_TEXT>            {0:M}
  +1 <<SOURCE_CITATION>>                        {0:M}
  +1 REFN <USER_REFERENCE_NUMBER>               {0:M}
    +2 TYPE <USER_REFERENCE_TYPE>               {0:1}
  +1 RIN <AUTOMATED_RECORD_ID>                  {0:1}
  +1 <<CHANGE_DATE>>                            {0:1}
```

Version 5.5.1

```
NOTE_RECORD :=
n @<XREF:NOTE>@ NOTE<SUBMITTER_TEXT>            {1:1}
  +1 [CONC|CONT] <SUBMITTER_TEXT>               {0:M}
  +1 REFN <USER_REFER ENCE_NUMBER>              {0:M}
    +2 TYPE <USER_REFERENCE_TYPE>               {0:1}
  +1 RIN <AUTOMATED_RECORD_ID>                  {0:1}
  +1 <<SOURCE_CITATION>>                        {0:M}
+1 <<CHANGE_DATE>>                              {0:1}
```

Anmerkungszeilen dürfen nicht mit einem Leerzeichen enden, da dieses vom Leseprogramm nicht erkannt werden kann. Der Schreiber kann aber die Aufspaltung in mehrere Zeilen erst nach der Umwandlung in einen der zugelassenen Codes vornehmen.

Mit dem Umstieg auf UNICODE in GEDCOM 5.5.5 sind jetzt auch signifikante Leerstellen an beiden Enden der Zeichenkette erlaubt.

2.5.5 Quellarchiv (repository)

Im Quellarchiv erfassen wir die Adressdaten des Archivs, in dem die Quellen abgelegt sind.

Version 5.5

```
REPOSITORY_RECORD :=
  n  @<XREF:REPO>@ REPO                     {1:1}
  +1 NAME <NAME_OF_REPOSITORY>              {0:1}
  +1 <<ADDRESS_STRUCTURE>>                  {0:1}
  +1 <<NOTE_STRUCTURE>>                     {0:M}
  +1 REFN <USER_REFERENCE_NUMBER>           {0:M}
    +2 TYPE <USER_REFERENCE_TYPE>           {0:1}
  +1 RIN <AUTOMATED_RECORD_ID>              {0:1}
+1 <<CHANGE_DATE>>                          {0:1}
```

Version 5.5.1

```
REPOSITORY_RECORD :=
n @<XREF:REPO>@ REPO{1:1}
+1 NAME<NAME_OF_REPOSITORY>                 {1:1}
+1 <<ADDRESS_STRUCTURE>>                    {0:1}
+1 <<NOTE_STRUCTURE>>                       {0:M}
+1 REFN <USER_REFERENCE_NUMBER>             {0:M}
   +2 TYPE <USER_REFERENCE_TYPE>            {0:1}
+1 RIN <AUTOMATED_RECORD_ID>                {0:1}
+1 <<CHANGE_DATE>>                          {0:1}
```

2.5.6 Quelle (source)

Eigentlich sollten alle Informationen durch Originalquellen belegt sein.

Version 5.5		Version 5.5.1	
SOURCE_RECORD :=		SOURCE_RECORD :=	
n @<XREF:SOUR>@ SOUR	{1:1}	n @<XREF:SOUR>@ SOUR	{1:1}
+1 DATA	{0:1}	+1 DATA	{0:1}
+2 EVEN <EVENTS_RECORDED>	{0:M}	+2 EVEN <EVENTS_RECORDED>	{0:M}
+3 DATE <DATE_PERIOD>	{0:1}	+3 DATE <DATE_PERIOD>	{0:1}
+3 PLAC <SOURCE_JURISDICTION_PLACE>	{0:1}	+3 PLAC <SOURCE_JURISDICTION_PLACE>	{0:1}
+2 AGNC <RESPONSIBLE_AGENCY>	{0:1}	+2 AGNC <RESPONSIBLE_AGENCY>	{0:1}
+2 <<NOTE_STRUCTURE>>	{0:M}	+2 <<NOTE_STRUCTURE>>	{0:M}
+1 AUTH <SOURCE_ORIGINATOR>	{0:1}	+1 AUTH <SOURCE_ORIGINATOR>	{0:1}
+2 [CONT|CONC] <SOURCE_ORIGINATOR>	{0:M}	+2 [CONC|CONT] <SOURCE_ORIGINATOR>	{0:M}
+1 TITL <SOURCE_DESCRIPTIVE_TITLE>	{0:1}	+1 TITL <SOURCE_DESCRIPTIVE_TITLE>	{0:1}
+2 [CONT|CONC] <SOURCE_DESCRIPTIVE_TITLE>	{0:M}	+2 [CONC|CONT] <SOURCE_DESCRIPTIVE_TITLE>	{0:M}
+1 ABBR <SOURCE_FILED_BY_ENTRY>	{0:1}	+1 ABBR <SOURCE_FILED_BY_ENTRY>	{0:1}
+1 PUBL <SOURCE_PUBLICATION_FACTS>	{0:1}	+1 PUBL <SOURCE_PUBLICATION_FACTS>	{0:1}
+2 [CONT|CONC] <SOURCE_PUBLICATION_FACTS>	{0:M}	+2 [CONC|CONT] <SOURCE_PUBLICATION_FACTS>	{0:M}
+1 TEXT <TEXT_FROM_SOURCE>	{0:1}	+1 TEXT <TEXT_FROM_SOURCE>	{0:1}
+2 [CONT|CONC] <TEXT_FROM_SOURCE>	{0:M}	+2 [CONC|CONT] <TEXT_FROM_SOURCE>	{0:M}
+1 <<SOURCE_REPOSITORY_CITATION>>	{0:1}	+1 <<SOURCE_REPOSITORY_CITATION>>	{0:M}
+1 <<MULTIMEDIA_LINK>>	{0:M}	+1 REFN <USER_REFERENCE_NUMBER>	{0:M}
+1 <<NOTE_STRUCTURE>>	{0:M}	+2 TYPE <USER_REFERENCE_TYPE>	{0:1}
+1 REFN <USER_REFERENCE_NUMBER>	{0:M}	+1 RIN <AUTOMATED_RECORD_ID>	{0:1}
+2 TYPE <USER_REFERENCE_TYPE>	{0:1}	+1 <<CHANGE_DATE>>	{0:1}
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	+1 <<NOTE_STRUCTURE>>	{0:M}
+1 <<CHANGE_DATE>>	{0:1}	+1 <<MULTIMEDIA_LINK>>	{0:M}

Quell-Datensätze werden benutzt, um eine bibliografische Beschreibung der zitierten Quelle zu liefern.

2.5.7 Einreichung (submission), Verfasser (submitter)

Der Einreichungs-Datensatz tritt nur einmal im Kopf der GEDCOM-Datei auf und wird in der Verfassertabelle abgelegt. Verweise auf Einträge in der Verfassertabelle sind nur notwendig, wenn es sich um eine abweichende Quelle handelt.

Version 5.5		Version 5.5.1	
SUBMISSION_RECORD :=		SUBMISSION_RECORD :=	
n @<XREF:SUBN>@ SUBN	{1:1]	n @XREF:SUBN@ SUBN	{1:1}
+1 SUBM @<XREF:SUBM>@	{0:1}	+1 SUBM @XREF:SUBM@	{0:1}
+1 FAMF <NAME_OF_FAMILY_FILE>	{0:1}	+1 FAMF <NAME_OF_FAMILY_FILE>	{0:1}
+1 TEMP <TEMPLE_CODE>	{0:1}	+1 TEMP <TEMPLE_CODE>	{0:1}
+1 ANCE <GENERATIONS_OF_ANCESTORS>	{0:1}	+1 ANCE <GENERATIONS_OF_ANCESTORS>	{0:1}
+1 DESC <GENERATIONS_OF_DESCENDANTS>	{0:1}	+1 DESC <GENERATIONS_OF_DESCENDANTS>	{0:1}
+1 ORDI <ORDINANCE_PROCESS_FLAG>	{0:1}	+1 ORDI <ORDINANCE_PROCESS_FLAG>	{0:1}
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	+1 RIN <AUTOMATED_RECORD_ID>	{0:1}
		+1 <<NOTE_STRUCTURE>>	{0:M}
		+1 <<CHANGE_DATE>>	{0:1}

Das sendende System nutzt einen „Einreichungs-Datensatz“, um dem empfangenden System Instruktionen und Informationen zu übermitteln. TempleReady verarbeitet diese, um zu bestimmen, zu welchem Tempel die freigegebenen Datensätze weitergeleitet werden. Der Submissions-Datensatz wird auch für die Kommunikation zwischen Ancestral File und TempleReady genutzt. Jede GEDCOM Übertragung muss genau einen Submissions-Datensatz enthalten. Mehrere Einreichungen werden in separate GEDCOM-Übermittlungen aufgetrennt.

Version 5.5		Version 5.5.1	
SUBMITTER_RECORD :=		SUBMITTER_RECORD :=	
n @<XREF:SUBM>@ SUBM	{1:1}	n @<XREF:SUBM>@ SUBM	{1:1}
+1 NAME <SUBMITTER_NAME>	{1:1}	+1 NAME<SUBMITTER_NAME>	{1:1}
+1 <<ADDRESS_STRUCTURE>>	{0:1}	+1 <<ADDRESS_STRUCTURE>>*	{0:1}
+1 <<MULTIMEDIA_LINK>>	{0:M}	+1 <<MULTIMEDIA_LINK>>	{0:M}
+1 LANG <LANGUAGE_PREFERENCE>	{0:3}	+1 LANG <LANGUAGE_PREFERENCE>	{0:3}
+1 RFN <SUBMITTER_REGISTERED_RFN>	{0:1}	+1 RFN <SUBMITTER_REGISTERED_RFN>	{0:1}
+1 RIN <AUTOMATED_RECORD_ID>	{0:1}	+1 RIN <AUTOMATED_RECORD_ID>	{0:1}
+1 <<CHANGE_DATE>>	{0:1}	+1 <<NOTE_STRUCTURE>>	{0:M}
		+1 <<CHANGE_DATE>>	{0:1}

Der Verfasser-Datensatz (SUBMITTER_RECORD) identifiziert eine Person oder Organisation, die die Informationen der GEDCOM-Übermittlung zusammengetragen haben. Es wird davon ausgegangen, dass alle Datensätze einer Übermittlung von dem im Dateikopf genannten Verfasser zusammengestellt wurden, es sei denn, es wird innerhalb einzelner Datensätze durch SUBM auf einen anderen Verfasser-Datensatz verwiesen.

2.6 Nachschlagtabellen (Unterstrukturen)

2.6.1 GEDCOM-Unterstrukturen

Nachschlagtabellen (in GEDCOM Unterstrukturen) sind Tabellen, von denen selbst keine Beziehungen auf andere Tabellen ausgehen. Sie enthalten keine Fremdschlüssel (Verweis-IDs), jedoch Primärschlüssel (Referenz-IDs) mit denen sie von anderen Strukturen angesprochen werden können.

ADDRESS_STRUCTURE	Vollständige Anschrift
ASSOCIATION_STRUCTURE	Verknüpfungen zwischen zwei Personen im Sinne einer Rolle, die im Unterelement RELA beschrieben wird.

Theoretisch ist die Assoziation die Idee eines ternären (bzw. noch höheren) Beziehungstyps, der zwei Personen und die Rolle dabei modelliert:

Eine 'Person' wird in der Rolle 'Hebamme' zu einer Person 'Kind‘ beim Ereignis 'Geburt' tätig.

Im Ansatz finden wir dazu bei GEDCOM folgenden „Rucksack“ (nachträgliche Ergänzung) in der Beschreibung der Primitiven:

ROLE_DESCRIPTOR: *= {Size=1:25}*
A word or phrase that identifies a person's role in an event being described. This should be the same word or phrase, and in the same language, that the recorder used to define the role in the actual record.

ROLE_IN_EVENT: *= {Size=1:15}*
[CHIL | HUSB | WIFE | MOTH | FATH | SPOU | (<ROLE_DESCRIPTOR>)]
Indicates what role this person played in the event that is being cited in this context. For example, if you cite a child's birth record as the source of the mother's name, the value for this field is „MOTH.“ If you describe the groom of a marriage, the role is „HUSB.“ If the role is something different than one of the six relationship role tags listed above then enclose the role name within matching parentheses.

Neben den vorgegebenen, eigentlich schon anderweitig besetzten Rollen können somit auch benutzerspezifische Rollen ergänzt werden. Diese Eigenschaften gehören zur Quellenbeschreibung und sollen die abgeleiteten Informationen näher beschreiben.

2.6.2 Weitere Nachschlagtabellen

Nicht nur durch die fehlende Ortstabelle sondern auch durch weitere fehlende Nachschlagtabellen kommt es an vielen Stellen einer GEDCOM-Datei zu Wiederholungen mit der Gefahr der Widersprüchlichkeit. Eine sicher nicht vollständige Liste könnte folgendermaßen aussehen:

Aliasname	mögliche Aliasnamen für Geburts- und Nachname
AliasVorname	Sammlung der verschiedenen Varianten zu einem Vornamen
Archiv	Sammlung der Archive
Beziehung	das im Text entwickelte Modell der Beziehungen zwischen zwei Personen
Beziehungsart	Ereignisliste für das Entstehen einer Beziehung
Beziehungsrolle	Rolle der Person in der Beziehung
Bild	Tabelle mit allen Einzelbildern
BildGruppe	Tabelle mit allen Gruppenbildern
DokumentKategorie	erlaubte Kategorien eines Dokuments
Hebamme	Variante zu den Beziehungen mit eigenständiger Tabelle
Herrschaft	Liste aller Herrschaften (bei Leibeigenschaft/Sklaverei
Land	Liste aller Länder
Nationalitaet	Sammlung der möglichen Nationalitäten
OrtGesamt	Orte
OrtHaus	Häuser in einer Straße

OrtStrasse	Straßen in einem Ort
Pfarrer	Tabelle mit den Pfarrern, die im Stammbaum aktiv werden
Religion	erlaubte Religionen

Die die im RDM notwendigen mx:mx-Zwischentabellen sind nicht aufgeführt. Alle Nachschlagtabellen können im Gegensatz zu Wertetabellen beliebig bearbeitet/erweitert werden.

2.7 Zusammenfassungen

Die Zusammenfassungen dienen der übersichtlicheren Darstellung der Grammatik. Viele der Elemente wird man kaum in GEDCOM-Dateien sehen. Sie selbst enthalten keine Schlüssel, können aber Elemente mit Schlüsseln enthalten. Die Details dazu müssen im Standard selbst nachgelesen werden.

CHANGE_DATE	Dem Datum der letzten Änderung werden Notizen hinzugefügt.
CHILD_TO_FAMILY_LINK (FAMC)	Zeiger Person->Familie in der Rolle ‚Kind von'
EVENT_DETAIL	
FAMILY_EVENT_DETAIL	
INDIVIDUAL_ATTRIBUTE_STRUCTURE	
INDIVIDUAL_EVENT_DETAIL	
INDIVIDUAL_EVENT_STRUCTURE	
LDS_INDIVIDUAL_ORDINANCE	
LDS_SPOUSE_SEALING	
MULTIMEDIA_LINK	
PERSONAL_NAME_PIECE	
PERSONAL_NAME_STRUCTURE	
PLACE_STRUCTURE	
SOURCE_CITATION	
SOURCE_REPOSITORY_CITATION	
SPOUSE_TO_FAMILY_LINK (FAMS)	Zeiger Person->Familie in der Rolle ‚Partner von' | ‚Elter in'

2.8 Zeichensatz und Verschlüsselung

Im Laufe der EDV-Geschichte hat sich eine Vielzahl von Zeichensätzen und Verschlüsselungen herausgebildet, die teilweise auf die Zeit vor der modernen EDV zurückgehen, als man die Zeichen für die grenzüberschreitenden Fernschreibdienste usw. vereinheitlichte. So ist auch der Morse-Kode eine Form der Zeichenverschlüsselung.

GEDCOM 5.5 kennt die Zeichensätze ASCII, ANSEL und UNICODE. Wobei ASCII und ANSEL sowohl den Zeichensatz wie auch die Verschlüsselung bezeichnen.

Bei GEDCOM 5.5.1 ist die Verschlüsselung UTF-8 hinzugekommen.

GEDCOM 5.5.5 schreibt die Verwendung von UNICODE als *Zeichensatz* und UTF-8 oder UTF-16 als *Verschlüsselung* vor (S. 47). ASCII und ANSEL sind nicht mehr zugelassen. Die Feldlänge wird nicht mehr in Zeichen (characters) sondern in Kodeeinheiten (code units) gemessen.

Der in der GEDCOM-Datei verwendete Zeichensatz wird im Kopf-Tag mit `CHAR` festgelegt. Version 5.5.5 legt den Wert `UNICODE` auf die Verschlüsselung `UTF-16` fest. In den westeuropäischen Sprachen reicht `UTF-8` aus.

2.8.1 ASCII

Der ASCII (American Standard Code for Information Interchange) ist wie der Name schon sagt, ein Austauschkode zwischen Maschinen. Der Zeichensatz benutzt sieben Bits pro Zeichen zur Verschlüsselung und ein achtes Bit zur Fehlererkennung. Mit sieben Bits können 2^7 = 128 Zeichen festlegt und mit den Ordnungsnummern 0 bis 127 verschlüsselt werden. Ziehen wir die für die Steuerung (der Datenübertragung mittels Fernschreiber) 33 notwendigen Bitkombinationen ab, so verbleiben 95 „darstellbare/druckbare“ Zeichen übrig. Für das lateinische Alphabet benötigen wir 2 × 26 = 52 Buchstaben und 10 Ziffern. Der Rest sind 23 Satz- und Sonderzeichen.

Ein Trick von ASCII besteht darin, dass sich Groß- und Kleinbuchstaben um den Wert 32 unterscheiden, so dass wir genau ein Bit ändern müssen, um zwischen den Zeichen zu wechseln. Dies ist bei allen später entwickelten Kodes für einen erweiterten Zeichensatz nicht mehr möglich. Hier müssen Umwandlungstabellen angelegt werden. So wurde im Deutschen inzwischen ein großes ß eingeführt.

Da wir in einem Computer diese Fehlererkennung nicht benötigen, wurde das achte Bit zur Erweiterung des Zeichensatzes benutzt. Dadurch entstanden die so genannten Code-Pages mit je 256 Bitkombinationen für verschiedene Sprachgemeinschaften (auch erweiterter ASCII genannt, was natürlich das A für American ad absurdum führt). Für Mitteleuropa war beispielsweise „Latin-1“ zuständig (ursprünglich von ANSI (American National Standards Institute, vergleichbar mit dem DIN) entworfen).

Die verschiedenen Code-Pages finden wir weitgehend in der Basic Multilingual Plane (BMP) von UNICODE wieder.

2.8.2 ANSEL

Der ANSEL (American National Standard for Extended Latin Alphabet Coded Character Set for Bibliographic Use) ist eine Erweiterung speziell für bibliografische Anwendungen. Neben einigen speziellen Buchstaben (wie das æ) und Zeichen erweitert er ASCII um 63 Diakritika mit der Absicht immer einen Buchstaben mit einem Diakritikon zu verschmelzen, also ¨ + a zu ä. Die Idee dahinter bestand darin, die lateinischen Buchstaben von den Diakritika zu trennen, um weiter mit nur 256 Symbolen auszukommen. Mechanisch wurde die Kombination bei den Schreibmaschinen dadurch realisiert, dass die Taste keinen Vorschub in der Zeile machte und auf das nächste Zeichen wartete. Das Diakritikon steht somit vor dem lateinischen Buchstaben. Auch auf den Computer-Tastaturen gibt es solche Tasten, die sich an die jeweils gewählte Sprache mit Hilfe geeigneter Treiber anpassen. Logischerweise sind damit aber alle anderen Schriftfamilien wie Russisch, Griechisch usw. ausgeschlossen, die nicht auf den lateinischen Buchstaben basieren und diese durch Diakritika abwandeln.

GEDCOM hat den ANSEL-Zeichensatz noch einmal abgewandelt und trotzdem weiter ANSEL genannt (also eigentlich LDS ANSEL). GEDCOM belegte einige Zeichen anders, da beispielsweise das ß fehlte und nicht durch Kombination erzeugt werden konnte. Weiterhin gibt es auch noch verschiedene ANSEL-Varianten pro GEDCOM Version.

Außerhalb von GEDCOM hat sich aber dieser Zeichensatz nie durchgesetzt, so dass er von einigen Genealogie-Programmen nicht verstanden wird. Bis GEDCOM 5.5.1 war dieser Zeichensatz der bevorzugte Standard, während die EDV schon lange auf UNICODE umgestiegen ist.

2.8.3 UNICODE

UNICODE ist ein Vier-Byte-Zeichensatz, der langfristig alle vorhandenen und auch neu erfundenen Zeichen kodieren soll. Nun benötigt man nicht immer die gesamte Länge, so dass man versucht hat, eine platz- und bei der Datenübertragung Bandbreite sparendere Variante zu entwickeln.

Derzeit sind 137.929 Zeichen (Version 12, März 2019) festgelegt (inklusive Domino- und Mah-Jongg-Steine und auch Emoji).

Der UNICODE wurde in mehreren Schritten und wird weiter von der ISO (International Standardization Organization) als UCS (Universal Coded Character Set) genormt. Seit 2011 ist UCS mit UNICODE (UTF-16) bzw. UNICODE (UTF-32) identisch. Davor gab es einige Unterschiede.

Eine Technik der Verkürzung besteht darin, nur die letzten zwei Bytes (16 Bit) von UNICODE zu nutzen, also nur 65.526 Zeichen. Um aber noch platzsparender zu kodieren, hat man sich mit UTF (*UCS Transformation Format)* ein variables Format überlegt.

UTF-8 (8-Bit *UCS Transformation Format*) entspricht in den ersten 128 Zeichen dem ASCII, so dass er von allen Programmen (Editoren) verstanden werden und mit jeder Tastatur erzeugt werden sollte.

Jedes UNICODE-Zeichen kann in bis zu vier Bytes von UTF-8 transformiert. Einerseits wollen wir mit den „alten“ Kodes kompatibel bleiben, andererseits aber den Zeichensatz erheblich ausweiten. Daneben soll die Lösung aber auch noch möglichst platz- und bandbreite-sparend bleiben. Daher müssen wir nur noch dafür sorgen, dass wir die häufig benutzen ASCII-Zeichen von den weiteren eher selten benutzen Zeichen unterscheiden können. Dies geschieht durch das erste Bit.

Alle ASCII-Zeichen (sieben Bits = 128 Zeichen) sind an einem 0-Bit an der ersten Stelle erkennbar, also

0000 0000 – 0111 1111

Das ist einfach. Jetzt wird es komplizierter. Da bisher nur ein Teil von UNICODE überhaupt belegt ist und auch davon nur ein Teil für unsere Schriftfamilien genutzt wird, transformieren wir diesen in einen neuen Kodebereich mit einem kleineren Zeichenvorrat ab.

Unicode-Bereich (4 Byte hexadezimal)	**UTF-8-Kodierung (binär, Schema)**	
0000 0000 – 0000 007F	0xxxxxxx	ASCII
0000 0080 – 0000 07FF	110xxxxx 10xxxxxx	es folgt ein Byte
0000 0800 – 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx	es folgen zwei Bytes
0001 0000 – 0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	es folgen drei Bytes

Beginnt das erste Byte mit einer 1, dann folgt eine Längenangabe durch die Anzahl der weiteren Einsen. Die Folgebytes beginnen immer mit 10, damit man sie auch „zwischendrin“ von den ASCII-Zeichen unterscheiden kann.

Wir sehen auch, dass die UNICODE-Zeichen ab `0x0011 0000` nicht nach UTF-8 umgesetzt werden können. In diesem oberen Bereich finden wir die asiatischen Zeichensätze, für die UTF-16 eingesetzt wird.

Um das Ganze noch ein weniger spannender zu machen, kann man die einzelnen Bytes auch noch in unterschiedlicher Reihenfolge in einer Datei ablegen oder an den Empfänger senden. Dazu wird die Bezeichnung „Dickender“ (big endian) und „Dünnender“ (little endian) eingeführt. Das sind Bezeichnungen, die aus dem Streit stammen, wie man ein Ei (vom dicken oder dünnen Ende her) zu essen habe (Gullivers Reisen). Daher gibt es UTF-16BE und UTF-16LE.

Für exotische Zeichensysteme wie Klingonisch steht ein privater Bereich (Private Use Area, PUA) bereit. Ehrlich gesagt, könnte man diesen auch für die Zeichenvielfalt einiger Kirchenbuchschreiber ganz gut nutzen.

2.8.4 Konvertierung

Ohne zu stark in die Details zu gehen, sollten wir uns die Probleme bei der Konvertierung zwischen den Verschlüsselungen klar machen.

Im Arbeitsspeicher liegen die Bytes ohne eigene Informationen nebeneinander. Erst durch das Programm werden Bytes bzw. eine Serie von Bytes interpretiert. Vereinfacht können wir annehmen, dass unser Prozessor ein Byte nach dem andern verarbeitet. Liest er das erste Byte von ANSEL, so muss er erkennen, ob es sich um ein Diakritikon handelt, zu dem er das nächste Byte mit dem Buchstaben lesen muss. Hat er nun das Zeichen, so muss er in der UNICODE-Tabelle das zugehörige UNICODE-Zeichen mit vier Bytes suchen. Jetzt erfolgt die Kompression nach UTF-8. Das Ergebnis kann jedoch nicht zurückgeschrieben werden, da der Speicherplatzbedarf unterschiedlich sein kann. Wir benötigen somit einen Ergebnispuffer für den transformierten Text.

Zum Glück müssen wir dies meist nicht selbst programmieren, da viele Programmiersprachen für diese Standardaufgabe entsprechende Funktionen bereitstellen.

2.8.5 BOM

BOM (Byte Order Mark, wie `U+FEFF`) ist eine Kennung, die ganz am Anfang einer GEDCOM-Datei stehen kann. Sie gibt die Reihenfolge der Bytes im nachfolgenden Text an. Die Angabe der Reihenfolge ist bei allen Verschlüsselungen notwendig, die aus mehreren Bytes bestehen (siehe auch 2.8.11 Anmerkungen).

Die Angabe zur Verschlüsselung wie `UTF-8` erscheinen im GEDCOM-Header erst nach vielen Bytes, kann also dafür nicht mehr ausgewertet werden. Daher verlangt GEDCOM 5.5.5 eine BOM am Anfang der Datei.

Auch wenn dadurch die Zeile `HEAD.CHAR` überflüssig wird, bleibt sie aus Kompatibilitätsgründen mit 5.5.1 weiter erlaubt.

2.8.6 Datum und Datumsformat

Schon im normalen Schriftwechsel stellt die Darstellung eines Datums in den verschiedenen Ländern ein Problem dar:

https://de.wikipedia.org/wiki/Datumsformat

https://en.wikipedia.org/wiki/Date_format_by_country

Die deutsche Norm DIN 5008 (Schreib- und Gestaltungsregeln für die Textverarbeitung) scheiterte 1996 mit ihrem Versuch, das in Deutschland übliche numerische Datumsformat (TT.MM.JJJJ) (in vorauseilendem Gehorsam) abzuschaffen, so dass diese Schreibweise in der Ausgabe 2001 wieder zugelassen wurde.

Eigentlich sollte man das Datumsformat anhand des Trennzeichens erkennen:

Punkt	`dd.mm.[jj]jj`	in Deutschland übliche numerische Datumsformat
Schrägstrich	`mm/dd/[jj]jj`	in angelsächsischen Ländern übliche numerische Datumsformat
Mittelstrich	`jjjj-mm-dd`	das aus der ISO 8601 abgeleitete, besonders empfohlene Sortierformat

GEDCOM verwendet ein verkürztes, englischsprachiges Datumsformat, indem es den Monat mit drei Buchstaben schreibt:

```
DATE [d]d MMM jjjj
```

```
TIME hh:mm:sss
```

Neben der englischen Monatsabkürzung mit drei Buchstaben erlaubt GEDCOM 5.5.5 auch eine vierbuchstabige (französische) und eine dreibuchstabige hebräische Variante.

Die fhiso (Family History Information Standards Organisation) hat nach vielen Entwicklungsjahren immerhin ein Papier zum Datum mit 52 Seiten herausgebracht, aber leider auch nicht mehr:

https://fhiso.org/TR/elf-dates.pdf

Aber einige Programme geben das Datum in recht willkürlicher Form aus.

Der Entwurf von GEDCOM X sieht ein ganz anderes, kompaktes Datumsformat vor:

```
±YYYY[-MM[-DD[Thh:[mm[:ss[±hh[:mm]|Z]]]]]]
```

(https://github.com/FamilySearch/gedcomx/blob/master/specifications/date-format-specification.md)

Von dort stammt auch die der Zusatz 2019 CE (common era, current era, christian era) statt AD (anno domini).

Was aber ist mit den häufig „unscharfen“ Datumsangaben wie „vor, nach, um, ca., etwa, zwischen“ usw.?

Das führt dazu, dass es keine Möglichkeit gibt, die Datumsangaben einfach zu sortieren und gegeneinander abzugleichen (Reihenfolgeverletzungen).

Ein einschlägiges Beispiel für das Problem stellt GEDBAS dar. Hier ist es nicht möglich, Lebensereignisse einer einzelnen Person in die richtige Reihenfolge zu bringen, indem man die Spalte sortiert. Daher ist die Sortierung für diese Spalte einfach abgeschaltet ist. Die Textsortierung hilft auch nicht weiter, da „Beerdigung“ mit B anfängt.

Oft kennen wir die Datumsangaben nur ungenau. GEDCOM lässt daher eine Vielzahl von Zusätzen zu, um ein Datum unscharf zu machen, meist `ABT` für ungefähr. Auch eine relative Angabe eines Zeitbereichs ist erlaubt wie `BEF` für einen Zeitbereich vor einem Datum oder `BET` für ein Datum zwischen zwei Datumsangaben.

Dabei gibt es ein nicht zu unterschätzendes Programmproblem, wenn das Datumsformat wie in MS Office-Anwendungen auch immer die Uhrzeit abspeichert. Das „einfache“ Datum beginnt immer mit der Uhrzeit `00:00:00`. Jeder exakte Datumsvergleich zwischen einem allgemein angegebenen Datum und einem mit Uhrzeit angegebenen Datum liefert daher `falsch`. In Zeitbereichen gehört der Endtag nicht dazu, da er erst um `23:59:59` endet. Das muss man natürlich auch auf alle anderen Varianten berücksichtigen `BET 1910 AND 1920` erfordert beim Einlesen eine Erweiterung auf `BET 01.01.1910 00:00:00 AND 31.12.1929 23:59:59`.

2.8.7 UTF-8

UTF-8 ist – wie bereits im vorherigen Kapitel dargestellt – eine komprimierte UNICODE-Verschlüsselung. Sie ist für GEDCOM 5.5.1 nicht als Standard vorgesehen, wird aber von vielen Genealogie-Programmen verarbeitet und ist im Grunde inzwischen ein „Praxis-Standard“.

2.8.8 Zeilenschaltungen

Ein Überbleibsel aus dem „Krieg der Betriebssysteme“ sind die unterschiedlichen Zeilenschaltungen (EOL, End of Line).

ASCII und alle darauf aufbauenden Zeichensätze haben zwei Steuerzeichen

- `Cr` Wagenrücklauf (carriage return)
- `Lf` Zeilenvorschub (line feed)

Anwender von mechanischen Schreibmaschinen kennen noch den Vorgang. Man konnte den Wagen ohne Zeilenvorschub zurückfahren, um beispielsweise Wörter zu unterstreichen oder die Nullen durchzustreichen. Diese Funktion gab es parallel dazu bei den Fernschreibern und später bei den Typenraddruckern. Heute wird ein Pixelbild erzeugt und an den Drucker geschickt.

Windows hat diese Technik beibehalten und setzt zwei Bytes pro Zeilenschaltung ein. Unix will Speicherplatz sparen und setzt nur `Cr` ein. Für das Betriebssystem des Mac(intosh) blieb nur noch `Lf` übrig. Öffnen wir eine unter Unix geschriebene Datei mit dem Editor von Windows, so erscheinen keine Zeilenwechsel.

GEDCOM 5.5.51 ist noch großzügiger und erlaubt neben der Kombination `CrLf` auch `LfCr`, ein „Zeichen“ aber zwei Bytes. Was passiert, wenn wir eine Windows-Datei mit einer Unix-Datei verlängern, also gemischte EOL-Zeichen in einer Datei haben? Dies wird von GEDCOM 5.5.5 verboten.

Version 5.5.5 unterscheidet nun `terminator` für GEDCOM-Zeilen von `newline` in den überlangen Kommentaren, die mit `CONT` verlängert werden müssen. Ein wenig merkwürdig liest sich

A GEDCOM writer should not pick the terminator appropriate for the platform it runs on, but the terminator appropriate for the destination system (`HEAD.DEST`). GEDCOM writers should use ␍/␊ when unsure which terminator to pick. Web applications should always use ␍/␊.

Das hört sich so an, als ob es keine allgemeine, sondern nur auf Sender und Empfänger abgestimmten GEDCOM-Dateien gibt.

2.8.9 Leerraum

Unter einem Leerraum (weiße Leerstellen, whitespace) (nicht „Leerstelle“!) verstehen wir Zeichenkodes, die als ein oder mehrere Leerstellen bei der Darstellung (un-)sichtbar bleiben. Typisch dafür ist die Leerstelle selbst, aber auch der Tabulator, die geschützte Leerstelle, Trennzeichen usw. Die Wirkung des Tabulators ist vom Empfänger abhängig, der die Tab-Stopps festlegt.

Daneben gibt es in der Textverarbeitung auch Zeichen, die Speicherplatz brauchen aber keine Anzeige erzeugen.

Leerräume (whitespaces) werden häufig zur Formatierung (Einrücken) eines Textes eingesetzt. Formatierungsassistenten wie bei den Programmiersprachen sind in GEDCOM nicht vorgesehen.

GEDCOM 5.5.5 verbietet nachfolgende, überflüssige Leerräume, erlaubt aber nachfolgende signifikante Leerräume (S. 41).

Führende Leerräume sollten vermieden werden, sind aber grundsätzlich erlaubt und werden beim Import einer Tag-Zeile überlesen. An anderen Stellen werden sie ganz verboten.

2.8.10 Fortsetzung mit CONC/CONT

Ein Mangel von GEDCOM besteht in den fehlenden Ende-Tags (die wir aus HTML oder noch strikter aus XHTML kennen), der sich insbesondere bei den Fortsetzungszeilen eines Textes über 255 Zeichen hinaus bemerkbar macht. Der Empfänger kann nie sicher sein, dass am Zeilenende von kürzeren Zeilen nicht doch noch eine Leerstelle ist. GEDCOM schreibt daher vor, dass die Zerlegung **vor** der Leerstelle erfolgen soll, so dass diese auf die nächste Zeile übernommen werden soll. Nachlaufende Leerstellen in einer Zeile sind also verbo-

ten. Andererseits startet dann die Forstsetzungszeile mit einer zusätzlichen Leerstelle nach dem Tag, der nicht entfernt werden darf. Den Fehlinterpretationen war damit Tür und Tor geöffnet.

Nichts einfacher als das, sprach der Programmierer. Ich stelle die Zeile mit dem notwendigen Präfix `2 CONT` in UTF-8 zusammen und untersuche das 253. Byte (EOL `CrLf` folgt dahinter), ob es ein Leerzeichens `hex 20` ist. In diesem Fall wende ich die Regel an. Aber leider gibt es einen kleinen, aber hinterhältigen Nebeneffekt, nämlich wenn dieses Byte der Anfang eines 2-Byte-Zeichens ist. Dann werden wegen der Ungleichheit die beiden Bytes in der GEDCOM-Datei getrennt. Schauen wir die Zeilen in einem Editor an, so erscheinen zwei Ersatzzeichen (wiederum abhängig vom jeweiligen Editor). Nun hängt es davon ab, wie der Empfänger die Zeilen verarbeitet. Konkateniert er sie auf Byte-Ebene und wandelt sie dann erst in den internen Kode (z. B. UTF-16) um, oder wandelt er erst um und konkateniert dann? Tatsächlich muss der Split-Algorithmus auch noch das Byte auf ein führendes 1-Bit prüfen, das ein UTF-8-Zeichen einleitet.

Aber auch das reicht noch nicht aus, da der umzubrechende Text auch mehrere Leerzeichen enthalten darf, die der Benutzer eingegeben hat. Also muss die Zeile weiter verkürzt werden, um das letzte Nicht-Leerzeichen zu finden. Was aber mit den vielen eliminierten Leerzeichen?

Ohne Verstoß gegen den GEDCOM-Standard kann die Fortsetzung auch nach weniger Zeichen erfolgen. Es ist nicht vorgeschrieben, die volle Länge einer GEDCOM-Zeile auszunutzen.

GEDCOM 5.5.5 will das Problem dadurch lösen, dass keine Zeichen beim Schreiben und Lesen entfernt werden. Die Zeilenlänge wird nicht mehr in Bytes sondern in Kodeeinheiten gemessen.

GEDCOM 5.5.5 Writer Best Practice

- Use operating system or programming library functions to make sure you split between characters, not inside characters
- Otherwise, split wherever is convenient; this GEDCOM file will be read by a GEDCOM 5.5.5 or later reader

GEDCOM 5.5.5 Reader Rules

- import each line value as-is
- do not trim trailing white space from any GEDCOM line or line value
- do not trim leading white space from any line value

2.8.11 Anmerkungen

Die Anzahl der veränderten Zeichen pro ASCII-Zeichen ist unterschiedlich. So gibt es sechs (mit Æ sogar sieben) veränderte Zeichen zu A, aber nur eines mit Ç zu C. Die „Romanisierung“ der diakritischen Zeichen ist somit nicht wie der ASCII-Trick der Groß-/Klein-Umschaltung trivial und mit dem Umschalten eines Bits erledigt. Wir müssen tatsächlich für jedes diakritische Zeichen ein zugehöriges ASCII-Zeichen festlegen.

Die Schriftarten Kurrent oder Sütterlin finden wir im UNICODE nicht, da die grafische (typografische) Darstellung eine Zeichens – die Glyphe – unabhängig von seinem linguistischen Wert ist. Ständig werden neue Schriftarten (fonts) entwickelt, die denselben Text unterschiedlich präsentieren, aber den Inhalt nicht verändern.

Zeichensatz	BOM
UTF-8	`EF BB BF`
UTF-16 Big Endian	`FE FF`
UTF-16 Little Endian	`FF FE`
UTF-32 Big Endian	`00 00 FE FF`
UTF-32 Little Endian	`FF FE 00 00`

Der Kopf-Tag `0 CHAR` mit der Angabe des Zeichensatzes erscheint für ein lesendes Programm relativ spät. Daher kann es bereits einige der davor liegenden Informationen in falscher Reihenfolge gelesen haben. Von einigen Programmen wird das BOM (Byte Order Mark) an den Anfang einer Datei geschrieben und ggf. auch dort erwartet. Andere Programme (Editoren) überlesen das BOM und zeigen es nicht an. Öffnen wir eine Datei mit BOM in einem Editor, der die Zeichen nicht unterdrückt, so sehen wir die Zeichenfolge `ï»¿`

Einige Editoren (z. B. NotePad) versuchen bei fehlendem BOM die Verschlüsselung zu analysieren, wobei der Erfolg aber nicht garantiert wird.

2.8.12 Nichttransparente Zeichen

Nichttransparente Zeichen sind Zeichen aus dem jeweiligen Zeichensatz, die für ein Programm eine Sonderfunktion darstellen und somit für die Steuerung eines Programmes genutzt werden. Diese Zeichen dürfen im normalen Text nicht auftreten.

Nun kann man sich fragen, wie ein genealogisches Programm auf ein nichttransparentes Zeichen reagiert, welches nicht an der erwarteten Stelle steht:

@	schließt Schlüssel ein
/	schließt Nachnamen im `NAME`-Tag ein

Der Schrägstrich ist beispielsweise interessant, wenn der Benutzer einen Nachnamen in verschiedenen Schreibweisen eingibt und dabei gewohnheitsgemäß als Trennzeichen den Schrägstrich benutzt.

3 Flache Tabellen

GEDCOM beschreibt flache Tabellen (Listen, flat tables), wie wir sie beispielsweise in Excel anlegen und verarbeiten. Diese Tabellen werden über die Verweise miteinander verknüpft.

Daher finden wir eine Reihe von Anwendungen, die hauptsächlich mit solchen flachen Tabellen arbeiten, sei es nur zur Dateneingabe oder auch zur Verwaltung des Stammbaumes. Flache Tabellen sind die Vorstufe zu den Relationalen Datenbanken (RDB).

3.1 Schlüssel (Primär-, Fremdschlüssel, Verweise)

Mit einem so genannten **Primärschlüssel** (RDM) wird ein Objekt (Person, Ort, Dokument) eindeutig in einer Anwendung identifiziert. Die GEDCOM-Primärschlüssel bestehen aus einem Kennbuchstaben und einer laufenden Nummer, also immer aus einer Zeichenkette. Diese werden mit `@` eingerahmt.

In einem Programm besitzt jedes Objekt keine oder eine eindeutige Speicheradresse. Diese verschwindet aber mit dem Programmende. Beim Herausschreiben (Persistieren) eines Objektes muss daher der Primärschlüssel ergänzt werden.

Ein **Fremdschlüssel** (Verweis) ist eine Spalte oder eine Spaltenkombination einer Tabelle, welche auf einen Primärschlüssel einer anderen oder der gleichen Tabelle verweist (RDM). Auch wenn Speicher heutzutage relativ billig sind, kosten solch kombinierten Schlüssel natürlich erheblichen Speicherplatz.

3.2 Eingabe

Interessant ist die Frage, wie der Benutzer an die Fremdschlüsselwerte kommt. Einen neuen Primärschlüssel (plus alle weiteren Werte) kann man in einer Excel-Tabelle hinten als neue Zeile einfügen. Die Fremdschlüssel müssen wir aber erst zusammensuchen, bevor wir den neuen Primärschlüssel (Datensatz) eingeben können.

Eine führende Makrosammlung zur Verarbeitung von genealogischen Daten in Excel-Tabellen finden wir unter

https://www.gedtool.de

Damit können GEDCOM-Dateien eingelesen werden, Dateien abgeglichen, verglichen und zusammengeführt werden, identische Personen gefunden werden, Personen gelöscht werden usw.

Aber die Anleitung, wie eine neue Person eingegeben werden kann, findet man vergeblich.

4 Datenbankentwurf

4.1 Strukturvorschlag

Wenn wir uns fragen, ob GEDCOM all unsere Wünsche abbilden kann, so müssen wir die Wünsche zumindest grob formulieren, wobei wir die EDV-spezifischen Elemente erst einmal nicht weiter betrachten wollen.

Andererseits wollen wir aber genauer hinschauen. Schließlich sammeln wir unsere Dokumente nicht in ungeordnet in einem einzigen Schuhkarton, vielmehr in Ordnern mit Trennblättern für einzelne Kapitel. Auch die Kirchenbücher haben meist eine zusätzliche Struktur. Orte bestehen aus Straßen und Häuser. Dazu müssen wir unterschiedliche Begriffe definieren

Personen	Person
Quellen	Schuber, Foliant, Ordner
	Buch, Kapitel
	Dokument, Seite, Page, p., Blatt, Folie, Fol.

Orte	Gesamtort
	Straße
	Haus mit Hausnummer
Medien	Portraits, Gruppenbilder, Filme, Töne

Abgesehen von den Bezeichnungen können wir jede dieser Hauptgruppen in Frage stellen. Sind die Medien nicht auch Quellen? Unterscheiden wir sie nur, weil sie in unterschiedlichen Formaten vorliegen?

Betrachten wir ein praktisches Beispiel:

Ein dicker, alter Kirchenfoliant, vielleicht noch eingeschoben in eine Hülle (Schuber), besteht aus oft mehreren Büchern, als da wären:

- Geburtsbuch
- Geburtsverzeichnis (Index, Katalog)
- Heiratsbuch
- Heiratsverzeichnis
- Sterbebuch
- Sterbeverzeichnis
- Einsegnungsbuch (Kommunion, Konfirmation)

Diese müssen nicht unbedingt alle vorhanden sein. Schon bei der Namensgebung sollten wir gleich an die spätere Sortierung denken. Da stört natürlich das Einsegnungsbuch die Ereignisreihenfolge. Es ist aber vergleichsweise selten. Mit der Einführung der Standesämter müssen wir auch noch Geburtsbuch und Taufbuch unterscheiden, die sich aber selten im gleichen Schuber befinden. Normalerweise finden wir somit in einem (genealogisch klassischem) Schuber bis zu sechs Bücher mit unterschiedlichen formalen und logischen Inhalten.

Auch wenn in Böhmen bis heute die Konskriptionsnummern (zur militärischen Rekrutierung erfunden) noch immer weitgehend eingesetzt werden, besuchen wir ein Haus in einer Straße in einem (größeren) Ort. Erneut finden wir wie bei den Quellen eine dreistufige Hierarchie, die für unsere Zwecke vollständig ausreicht.

Natürlich ist eine Erweiterung nach „oben“ denkbar. Ein Ort befindet sich in einem Kreis, ein Kreis in einem Bezirk, ein Bezirk in einem Bundesland, ein Bundesland in einem Staat, ein Staat auf einem Kontinent, ein Kontinent auf einem Planeten, ein Planet in einem Sonnensystem, …

Der Fantasie sind keine Grenzen gesetzt, aber wir sehen, dass sich sowohl die Hierarchie wie auch die zeitliche Zuordnung ändern können. Ab 1881 finden wir in den westlichen Ländern neue, straßenbezogene Hausnummern. Auch können wir verschieden Hierarchien entwickeln: Politisch, verwaltungstechnisch, kirchlich, archivarisch.

Beim Systementwurf verwendet man dazu den Begriff der „Granularität“ (Körnigkeit): Wie fein untergliedern wir das System bzw. welche groben Oberbegriffe bilden wir? Diese Hierarchiebildung führt zu einfachen 1:m-Beziehungstypen.

Ein Ort besteht aus mehreren Straßen.	Eine Straße befindet sich in einem Ort.
Eine Straße besteht aus mehreren Häusern.	Ein Haus steht in einer Straße.

Diese Technik lässt sich auch auf „Streugemeinden“ übertragen, wie wir sie beispielsweise im Böhmerwald finden. Dort geben Forscher Orte mit sehr lückenhaften Hausnummern an. Die Hausnummern wurden in entsprechenden Gemeinden vergeben, die über ein Art „Bauamt“ verfügten. Diese haben die fortlaufenden Hausnummern über das gesamte Gemeindegebiet verteilt.

Ein Ort besteht aus mehreren Unterorten.	Ein Unterort gehört zu einer Gemeinde.
Die Gemeinde vergibt Hausnummern an die Unterorte.	Ein Haus gehört zu einer Gemeinde.

In diesem Fall ist die Hausnummer für jede Gemeinde eindeutig.

Die Granularität umfasst automatisch den Aspekt der „Verdichtung“, indem zwischen den Ebenen nur noch Zusammenfassungen (Summen, Mittelwerte, Spitzenwerte usw.) weitergereicht werden.

4.2 Invarianz und Historie

Wenn wir die Felder in den Tabellen anlegen, müssen wir uns über deren Inhalt klar werden.

Unter der Invarianz verstehen wir die Unveränderlichkeit von Daten über ihren gesamten Existenz-Zeitraum. Bei unserer wichtigsten Klasse `Person` könnten wir alle Merkmale speichern, die sich im Laufe des Lebens einer Person nicht ändern. Hier hilft ein Blick in die Kriminologie bzw. die moderne Personenerkennung. Leider steht dem Genealogen die Verfügbarkeit der Merkmale wie Fingerabdruck usw. meistens nicht zur Verfügung. Nur von hochrangigen Personen werden Locken oder ganze Skelette aufbewahrt.

Schon der `Name` kann sich während der Lebenszeit ändern, sei es durch Vaterschaftsanerkennung, Adoption, Gerichtsbeschluss usw. Inzwischen geht der Trend zur DNS-Analyse, bei der wir unsere und damit die Erbanlagen unserer nahen Verwandten in fremde Hände geben.

Je weiter wir in die Vergangenheit vordringen, desto variabler werden die Schreibweisen der Namen, die zu jener Zeit nur mündlich weitergegeben wurden. Welchen Namen benutzen wir? Führen wir einen einheitlichen Suchnamen (Sippennamen) ein und ergänzen ihn durch den Originalnamen, den wir entziffern?

Nicht nur die Eigenschaften der Personen sondern auch die Eigenschaften aller anderen Elemente unserer Datenbank können sich ändern.

Ortsnamen verändern sich im Laufe, die politische und verwaltungstechnische Struktur wird angepasst usw. Unter Historie verstehen wir das Aufbewahren früherer Versionen einer Eigenschaft, vorzugsweise mit der Angabe des Zeitbereichs.

Die Historie eines Merkmals erfordert eine Tabelle mit den entsprechenden Zeitdaten.

4.3 Ereignisorientierter Entwurf

Der genealogische Forscher beruft sich auf *Artefakte* (Überbleibsel aus der Vergangenheit). Diese entstehen bei *Ereignissen*. Alle anderen Artefakte wie bereits aufbereitete Zusammenstellungen, Erinnerungen, Anekdoten usw. sollten möglichst auf deren Ursachen hin aufgedröselt werden.

Es gibt Hauptereignisse, die immer im Leben einer Person auftreten:

- Zeugung
- Geburt
- Tod

und Nebenereignisse, die auftreten können:

- Taufe
- Einsegnung
- Heirat
- Geburt von Nachkommen
- Wohnungswechsel
- Fähigkeitserwerb
- usw.

Bei jedem der Ereignisse sind Personen als Hauptpersonen und Nebenpersonen beteiligt. Auch hier können wir die gerade angedachte Struktur übernehmen:

- Zeugung: Hier wird es schon aufgrund der medizinischen „Fortschritte“ schwierig, die Beteiligten eindeutig festzulegen, geschweige denn, den genauen Zeitpunkt zu bestimmen.
- Geburt: Hier ist immer die Mutter und das Kind beteiligt, auch wenn wir diese nicht unbedingt kennen.

Im Grunde können wir für alle Ereignisse gedanklich Sonderfälle erzeugen (natürlicher/unnatürlicher Tod). Aber eine Datenbank kann nicht alles abbilden. Sie ist immer nur ein *vereinfachtes Modell der Wirklichkeit*.

Bei jedem Ereignis spielen also Personen eine *Rolle*. Wir müssen nun bei der Modellierung unserer Datenbank entscheiden, ob diese Rolle eine Muss- oder Kann-Rolle ist. Dabei können auch technische Aspekte der Umsetzung in ein Programm einfließen:

- Wollen wir die biologischen Eltern gleich bei ihren Kindern verankern oder nur als eine der Rollen begreifen?
- Sollen die weiteren Personen (z. B. Hebamme, Pfarrer) zusätzlich gespeichert werden und mit ihrer Rolle an das Ereignis angehängt werden, oder gehören sie zur großen Menge unserer möglichen Vorfahren?

- Soll das direkte Dokument (z. B. Geburtseintrag) zu einem Ereignis bei jeder Person verankert werden oder als eine der Rollen eines Dokuments zu einem Ereignis gespeichert werden?

Mit anderen Worten: Es gibt (fast) beliebig viele Entwürfe, die sich mehr oder minder gut für die Verarbeitung eignen. Zur Erinnerung: GEDCOM hat nur einen davon ausgewählt. Betrachten wir ausschließlich die Fremdschlüssel in den Datensätzen, so finden wir:

- Eine Person „kennt“ nicht ihre Eltern.
- Eine Person kennt nicht ihre Kinder.
- Eine Person kennt ihre Familie(n).
- Eine Person kennt ihre Rolle in den Familien (Kind, Elter).
- Eine Familie kennt Eltern und Kinder.

Fehler beim Datenbankentwurf führen dazu, nachträglich so genannte *Rucksäcke* auf den Entwurf drauf zu packen. Rucksäcke führen aber wie in der Realität zu einer starken Belastung des Systems. Sie lassen sich nicht immer widerspruchsfrei (redundanzarm) hinzufügen.

- Hebammen und Pfarrer (zumindest die evangelischen) können auch als Eltern auftreten. Sollten sie also in einer getrennten Datei gespeichert werden oder doch in der Personentabelle?
- Andererseits wissen wir über diesen Personenkreis oft noch weniger als über unsere eigenen Ahnen.

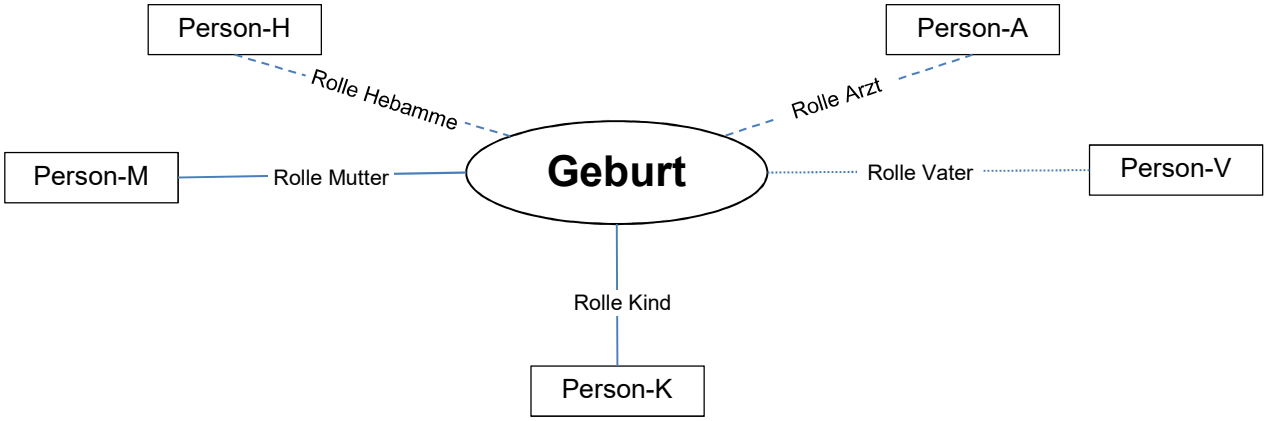


Abb. 3: Geburt, alles entsteht

An der Geburt (Abb. 3) sind Mutter und Kind (mindestens) beteiligt. Der Vater leitet sich aus der Befruchtung ab. Hebamme und/oder Arzt sind hilfreich, aber nicht unbedingt notwendig.

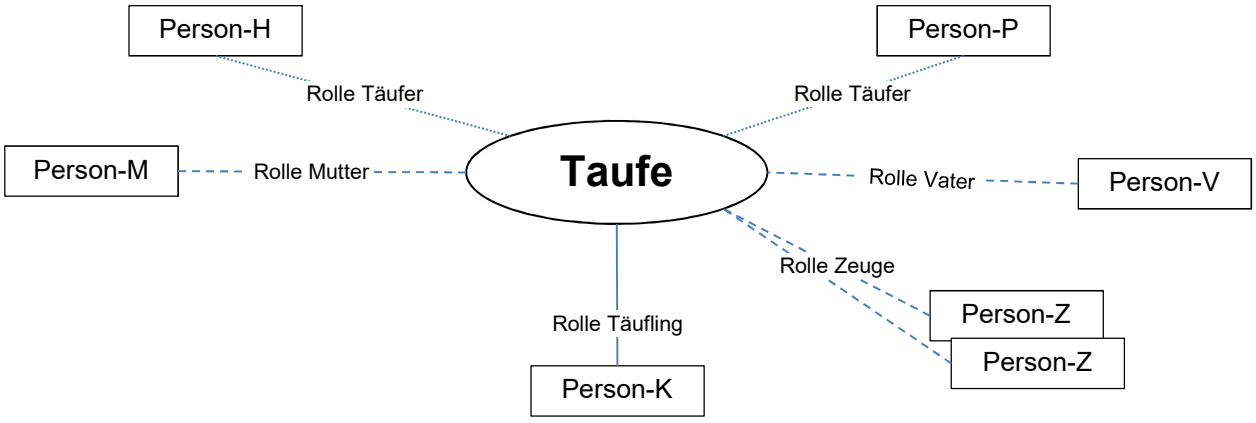


Abb. 4:Taufe, ein erster formaler Akt

Bei der Taufe (Abb. 4) müssen der Täufling und ein Täufer anwesend sein. Praktisch fehlt bei den historischen (Säuglings-) Geburten die Mutter, die noch in den ersten Stunden im Wochenbett liegt, während die Taufe möglichst zeitnah durchgeführt wird. Auch die Hebamme konnte eine Notgeburt durchführen. Sie war dann oft in einer zweiten Rolle als Zeugin anwesend.

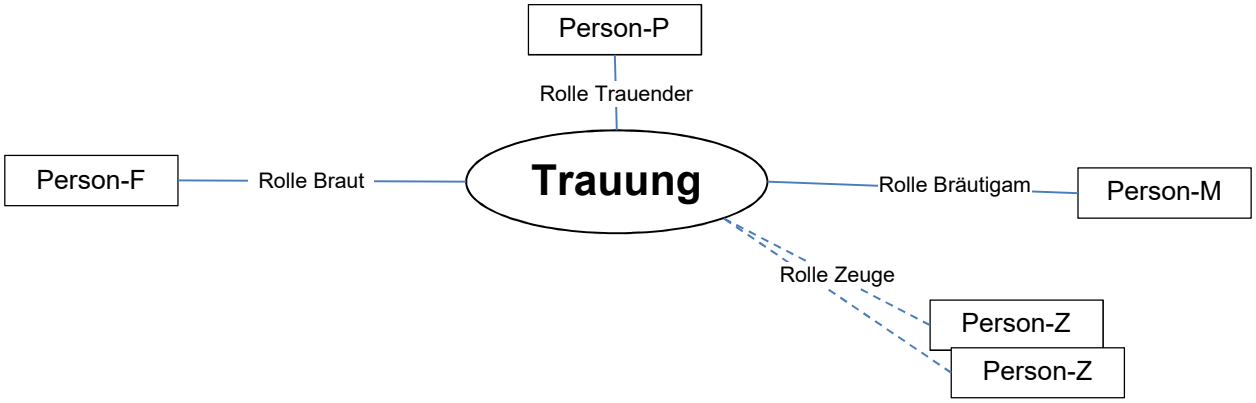


Abb. 5: Trauung

Neben den Brautleuten brauchen wir den Trauenden (Abb. 5), der Pfarrer, Standesbeamter, ja sogar Kapitän sein konnte. Üblicherweise kommen die Trauzeugen (Beistände) hinzu.

Für die Kernfunktionen unseres ereignisorientierten Entwurfs benötigen wir also nur das Ereignis, die Personen und deren Rollen, genauer Rollenbezeichnungen.

4.4 Kernentwurf I

Zentrale Aufgabe der Datenbank besteht darin, die Beziehungen zwischen den Personen abzubilden. Dazu kommen Zusatzinformationen wie Dokumente, Orte, Archive usw. Wir wollen uns erst einmal auf den Kernentwurf der Beziehungsarten konzentrieren.

Bitte beachten Sie, dass es sich um einen „familienlosen“ Datenbank-Entwurf handelt, der um eine Reihe von zusätzlichen Regeln ergänzt werden muss.

4.4.1 Allgemeiner Beziehungstyp

Zwischen zwei Personen herrscht in unserem Untersuchungsbereich ein rekursiver mx:mx-Beziehungstyp.

Eine Person hat möglicherweise zu mehreren anderen Personen eine Beziehung.	0…m
Eine Person hat keine Beziehung zu sich selbst.	
Eine Person hat möglicherweise keine Beziehung zu einer zweiten Person.	Einsiedler
Alle denkbaren Beziehungen sind vorerst optional.	
Die Beziehung entsteht durch ein Ereignis (oft leider nicht genau bestimmbar).	0…1
Die Beziehung endet durch ein Ereignis (oft leider nicht genau bestimmbar)	0…1

Ein mx:mx-Beziehungstyp wird in einem RDM mit Hilfe einer Zwischentabelle realisiert. Bei unserem reflexiven Beziehungstyp entstehen Verweise auf dieselbe Tabelle, nämlich `tblPerson`. Die Zwischentabelle ist ein **assoziativer Beziehungstyp**, da ihre Einträge erst durch ein **Ereignis** entstehen. Die beteiligten Personen spielen dabei unterschiedliche **Rollen**. Eine Person kann in der Zwischentabelle mit unterschiedlichen Rollen auftreten.

Rolle 1	**Rolle 2**	**Ereignis**
Sohn/Tochter	Mutter	Geburt
Sohn/Tochter	Vater	Geburt
Säugling	Hebamme	Geburt
Säugling	Arzt	Geburt
Kind	Mutter	Taufe
Kind	Vater	Taufe

Täufling	Pate (levans)	Taufe
Täufling	Pate (testes)	Taufe
Täufling	Pfarrer	Taufe

Durchaus diskussionsfähig ist die Festlegung der Rollenbezeichner. „Kind“ ist in der Umgangssprache ein Sammelbegriff mit vielen Schattierungen. Soll es die biologische Abstammung oder das Alter einer Person bezeichnen?

Die Rollenbezeichner verändern sich mit der Zeit. Waren in den alten Kirchenbüchern die Paten/Zeugen (insbesondere der Anzahl) wichtiger als die Eltern, der Namenstag wichtiger als der Geburtstag, der Vorname wichtiger als der Nachname usw., hat sich das im Laufe der Zeit stark verändert.

Weiterhin sind zusätzliche Regeln notwendig, die vom Tabellenentwurf nicht abgedeckt werden können (wie wir sie auch bei GEDCOM finden), beispielsweise:

Ein Kind kann höchstens eine biologische Mutter haben.
Ein Kind kann höchstens einen biologischen Vater haben.

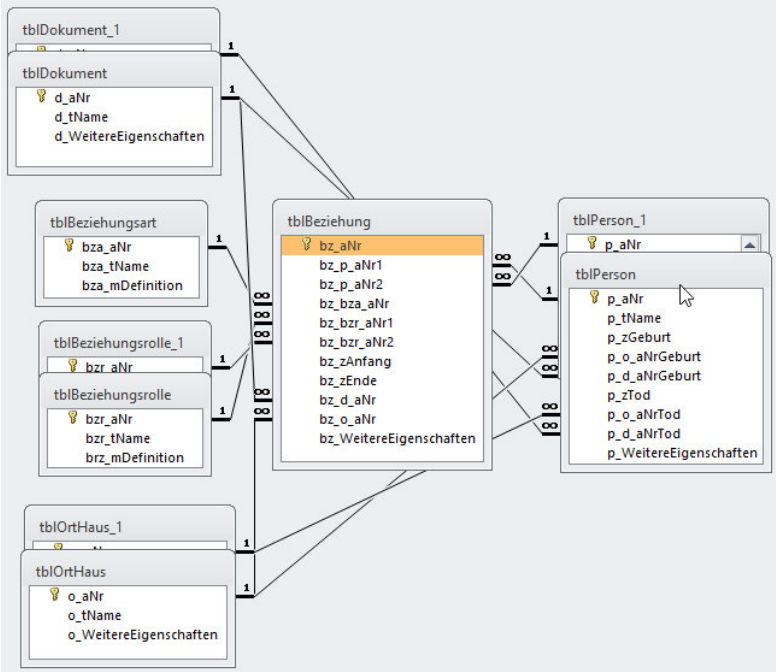


Abb. 6: Datenbank-Kernentwurf I ohne GEDCOM-Familie

Wir beginnen mit folgenden „physikalischen“ Tabellen, deren Datensätze man „anfassen“ kann (Abb. 6). Es sind „Nachschlagtabellen“, die vielfältig verknüpft sind, d. h., ihre Primärschlüsselwerte treten in vielen anderen Tabellen als Fremdschlüsselwerte auf.

Tabelle	**Beschreibung**
`tblPerson, tblPerson_1`	Eigenschaften der Personen/Individuen, insbesondere die Eigenschaften, die ohne eine zweite Person entstehen (z. B. Tod)
`tblDokument`	Eigenschaften der Dokumente/Artefakte/Bilder
`tblOrtHaus`	Eigenschaften des Ortes/Hauses

Wird eine Tabelle mehrfach verknüpft, so erhält sie einen Aliasnamen mit angehängter Nummer.

Anschließend fügen wir die „logischen“ Tabellen ein, die zur Darstellung der Beziehungen notwendig sind.

Tabelle	**Beschreibung**
`tblBeziehung`	Eigenschaften der Beziehung zwischen zwei Personen
`tblBeziehungsart`	Bezeichnung der Beziehung (hauptsächlich: Ereignis)

tblBeziehungsrolle	Bezeichnung der Rolle einer der Personen in der Beziehung

Die Beziehungen ergeben sich folgendermaßen:

- Eine Person hat bei einem Ereignis eine Beziehung zu einer zweiten Person in einer bestimmten Rolle.
- Dies gilt umgekehrt für eine zweite Person.
- Die Beziehungsart benötigt immer genau zwei (unterschiedliche) Personen.
- Das Ereignis wird durch ein Artefakt (Dokument, Bild, Erzählung, usw.) beschrieben.
- Ein Artefakt kann mehrere Ereignisse beschreiben.
- Das Ereignis findet an einem Ort statt.
- Ein Ort erfährt viele Ereignisse.
- Auch einzelpersonenbezogene Ereignisse finden an Orten statt.
- Auch einzelpersonenbezogene Ereignisse sind aus Dokumenten belegt.

4.4.2 Ereignisse vs. Zustand

Die Wahl der Eigenschaften ist diskussionsbedürftig. Soll zu einer Beziehung auch deren Ende erfasst werden? Grundsätzlich stellt ein Ereignis eine Momentaufnahme dar. Die Erfassung von Anfang und Ende ist die Beschreibung eines Zustandes. Dieser ergibt sich aber zwangsläufig aus zwei getrennten Ereignissen. Schauen wir uns das etwas genauer anhand von „Zuständen“ an:

Ereignis	**Art**	**Anmerkungen**
Geburt	Einmalereignis	
Mutter	Dauerzustand mit der Geburt	Beziehung
Vater	Dauerzustand mit der Geburt	Beziehung
Leben	Zustand zwischen Geburt und Tod	
Ehe	Zustand zwischen Trauung und Scheidung/Tod eines Partners	Beziehung
Trauung	Einmalereignis (pro Ehe)	Beziehung
Scheidung	optionales Einmalereignis (pro Ehe)	Beziehung
Tod	Einmalereignis	beendet u. U. Ehe

Wenn wir dazu noch von unscharfen Daten ausgehen, so werden die Regeln zur ständigen Aktualisierung von Anfang und Ende sehr schnell unübersichtlich.

Ist es nun sinnvoll, ständig (a priori) Anfang und Ende zu pflegen, oder sollte erst bei einer Abfrage (a posteriori) im Nachhinein die beiden Ereignisse eines Zustandes zusammengeführt werden? Der Zustand „Ehe“ lässt sich über die Suche nach der Trauung, der Scheidung oder dem Tod eines der Partner bestimmen.

Betrachten wir die Fremdschlüssel im GEDCOM-Record `individuals`, so finden wir keinen direkten Bezug `{0:1}` auf eine weitere Person. Vielmehr handelt es sich um mehrwertige Beziehungen `{0:M}`. Alle Beziehungsarten sind somit in der Tabelle `tblBeziehung` zusammenfassbar.

4.4.3 (A)Symmetrie

Die Einführung einer Beziehungstabelle sieht auf den ersten Blick genial einfach aus. Der Teufel steckt aber wie immer im Detail. Stellen wir einfach einmal folgende Frage:

„Welche Beziehungen hat eine Person zu anderen Personen?“

Sofort stellen wir fest, dass der Entwurf eine „Richtung“ zwischen Person 1 und Person 2 festlegt. Suchen wir nur in der ersten Spalte nach der Person, dann fehlen die Beziehungen, die wir in der zweiten Spalte eingetragen haben. Die Art der Beziehung zwischen zwei Personen ist asymmetrisch, je nach dem gewählten Standort.

- Eine Person hat einen Paten.
- Eine Person ist Pate einer zweiten Person.

Sollten wir nun beide Beziehungen eintragen, oder wäre es nicht besser, die Treffer bei der Suche in beiden Spalten zu vereinen?

Person 1	Person 2	Ereignis	Rolle 1	Rolle 2
P1	P2	Taufe	Täufling	Pate
P2	P1	Taufe	Pate	Täufling

Müssen wir also entweder beide Einträge erzwingen oder beide Einträge verhindern?

4.4.4 Reflexive Beziehungstypen

Von einem reflexiven (selbstbezüglichen) Beziehungstyp sprechen wir, wenn es sich um die Beziehungen zwischen zwei Entitäten (Elemente) desselben Entitätstyps (Tabelle) handelt. Wir haben im vorherigen Kapitel die Beziehung zwischen zwei Personen in unserem Stammbaum kennen gelernt, indem wir mit Hilfe einer Nachschautabelle `Rolle` die Beziehung näher beschreiben. Damit decken wir sowohl den (Stamm-) Baum als auch das Netz (allgemeinste Struktur) ab.

Wir können nun dieses Prinzip auch auf hierarchische Strukturen ausweiten, beispielsweise auf `Staat, Bezirk, Kreis, Ort, Straße, Haus, Etage, Wohnung` usw., indem wir die Beziehung allgemeine Objekte untereinander wieder mit Hilfe einer Nachschautabelle beschreiben.

Wir finden aber noch weitere Ansatzpunkte für reflexive Beziehungstypen `Dokument, Buch, Schuber, Raum, Archiv` usw.

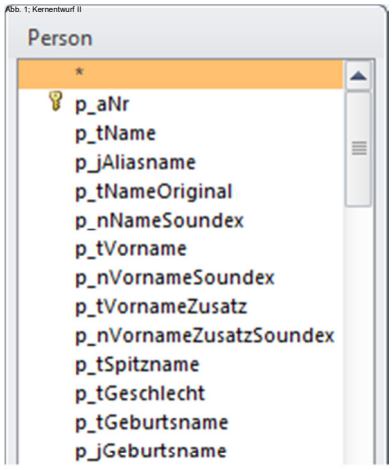
Speichern wir nun alle Elemente, außer den Personen, in einer einzigen Tabelle, oder legen wir für bestimmte Kategorien wie Dokumente, Orte getrennte Tabellen an?

GEDCOM 5.5.1 definiert zwar eine Eigenschaft `ROLE_IN_EVENT` mit stark eingeschränkten Werten, die aber lediglich in der Struktur `SOURCE_CITATION` bei der Quellenangabe zu einem Ereignis verwendet wird. Erweiterte, nichtfamiliäre Beziehungen sollen mit dem `ASSO`-Tag abgebildet werden.

4.4.5 GEDCOM und die freien Beziehungstypen

Dass GEDCOM feste Beziehungstypen zu anderen Tabellen herstellt, erkennen wir leicht an Fremdschlüsseln in den einem Tag nachgestellten Referenzen `@<XREF:…>@`. Schauen wir auf die freien Beziehungstypen, so erkennen wir zeitlich „nachgeschobenes Flickwerk“ (Rucksäcke), das von den verschiedenen Programmen unterschiedlich behandelt wird. Genau genommen handelt es sich nur bedingt um eine allgemeine Lösung. Vielmehr berücksichtigt GEDCOM beispielsweise primär die Rolle „Zeugen“ mit besonderen Tags. Aber ganz so schlecht wollen wir GEDCOM nicht machen, denn dazu wurde die `<<ASSOCIATION_STRUCTURE>>` eingeführt, die aber große Veränderungen zwischen den Versionen 5.3 und 5.5 erfahren hat. Diese Struktur verknüpft ausschließlich Personen mit Personen.

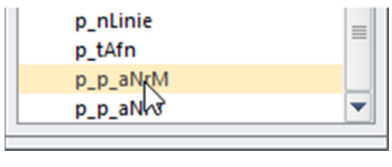
Die Mischung von freien und festen Assoziationen führt automatisch zu Ungereimtheiten. So können wir einen Zeugen beispielsweise sowohl mit dem Tag `ASSO` als auch mit dem `WITN` (Verwitwung) angeben. `WITN` wurde inzwischen wieder abgeschafft.



4.5 Kernentwurf II

Eine Variante des Kernentwurfs I besteht darin, neben den beliebig freien Rollen, einige gesichert, feste, invariante Rollen (wie Vater und Mutter, Braut und Bräutigam usw.) einzuarbeiten. Dieser Entwurf kommt GEDCOM schon näher, das die Familie als zentrale Tabelle ansieht, zu der Vater, Mutter und Kinder gehören.

In nebenstehendem Entwurf sehen wir die Eltern als Fremdschlüssel auf die Personentabelle selbst ganz am Ende der Felder, jeweils in der Rolle Mutter bzw. Vater. Betrachten wir die Fremdschlüssel in diesem Entwurf, dann können wir sagen:



- Eine Person kennt ihre Eltern.
- Die Eltern kennen aber nicht ihre Kinder.

Erst die Abfrage „Welches Kind hat meinen Primärschlüssel?“ kann ein Elter seine Kinder bestimmen. Diese müssen nicht dieselben sein, die sich bei

der Suche mit dem anderen Elter ergeben. Damit können wir „moderne Patchwork-Familien“ gut abbilden.

Natürlich sind in diesem Entwurf weitere Fremdschlüssel auf Orte und Dokumente vorhanden aber nicht sichtbar

4.5.1 GEDCOM und die festen Beziehungstypen

Dass auch GEDCOM feste Beziehungstypen zu anderen Tabellen herstellt, erkennen wir leicht an den Fremdschlüsseln. Einem Tag werden dazu Referenzen `@<XREF:…>@` nachgestellt. Für unseren ganz einfachen Kernentwurf sind dies die nachgestellten `@<XREF:INDI>@`-Referenzen. Die Multiplizität (Vielfachheit) in geschweiften Klammern gibt dann an, wie oft die Beziehungen möglich sind.
Multiplizität = Kardinalität {1:M} plus Konditionalität {0:x}.

Die biologische Vererbung garantiert, dass jede Person biologische Eltern (Personen) hat, auch wenn diese möglicherweise unbekannt sind. Das ist ein doppelter (getrennter) x:mx-Beziehungstyp (x={0..1}, mx= {0..m}) auf einen männlichen Vater und eine weibliche Mutter, genauer ein **reflexiver** Beziehungstyp auf dieselbe Tabelle.

4.6 Erweiterungen

Den Kernentwurf erweitern wir nun um viele weitere Objekte wie

- Dokumente
- Orte
- Bilder
- Geschichten
- Archive

Der Fantasie sind keine Grenzen gesetzt.

4.7 Was kann GEDCOM abbilden?

Diese Frage bezieht sich auf „natürliche“ Abbildung (benannte Eigenschaften), also nicht auf Konstrukte, wünschenswerte Daten irgendwo unter einem anderen Titel (meist als Kommentare) zu verstecken. Ehrlich gesagt ist es einfacher zu fragen: Was kann GEDCOM nicht abbilden?

1. Orte suchen wir vergeblich. Es gibt keinen entsprechenden Datensatz.
2. Eine Hierarchie (Zusammenfassung redundanter Daten) ist nirgends vorgesehen.

Natürlich können wir durch Textsuche beispielsweise die Orte finden. Es wird aber vom Entwurf her nicht sichergestellt, dass die Orte einheitlich geschrieben werden. Dokumente werden nicht in größeren Einheiten zusammengefasst usw. usw.

4.8 Ausblick auf GEDCOM X

An dieser Stelle ist auch ein Blick auf die Spezifikationen des GEDCOM X-Vorschlags interessant

http://www.gedcomx.org/Specifications.html

die wir auf

https://github.com/FamilySearch/gedcomx/blob/master/specifications/conceptual-model-specification.md

finden.

Hier kann nur ein kurzer Überblick über die 30 Datentypen (Tabellen) gegeben werden. Details und Beispiele sollten auf den Quellen nachgeschaut werden. Viele der Kapitel zu den Datentypen enthalten je ein Unterkapitel „Bekannte xxxx Typen“, die als Beispiele für die Implementation genutzt werden können.

Der Begriff „**Datentyp**“ entspricht der formalen Beschreibung einer Datenstruktur. Er stammt aus der OOP und kann als Klasse oder im RDM als Datensatzbeschreibung einer Tabelle angesehen werden. Ein Datentyp besteht aus der Aufzählung der Eigenschaften (`properties`) mit dem zugehörigen „Datentyp“ (Zahl, Text, Liste, usw.). Ein Datentyp (Datensatz) mit realen Werten wird zu einer **Instanz**.

Der Uniform Resource Identifier („URI“) ist für das konzeptionelle Modell von GEDCOM X von grundlegender Bedeutung. Der URI wird verwendet, um sowohl die Datentypen als auch die Dateninstanzen zu identifizieren.

Der URI wird auch verwendet, um Elemente der von GEDCOM X definierten kontrollierten Vokabulare zu identifizieren. Der URI wird in RFC (Request for Comments) 3986 spezifiziert.

URI als Datentyp eines Elements können wir als Link zu der jeweiligen Definition ansehen.

Eine weitere Quelle für ein mögliches XML-basiertes Dateiformat finden wir bei

> http://www.gaenovium.com/presentations/2014/Gaenovium 2014 - Michel Brinckman - The A2A Data Model and its application in WieWasWie.pdf

Es sieht vier Tabellen `Person, Event, Object, Source` vor, die über Beziehungen miteinander verknüpft sind. Die offensichtlichen Beziehungstypen (RDM-Zwischentabellen) lassen sich anhand der Namen erkennen `RelationEP, RelationPO, RelationEO`. In den zwei Nachschlagtabellen `RelationP, RelationO` werden die Art der Beziehungen zwischen Personen bzw. Objekten abgelegt.

Interessant sind die beiden Tabellen für die Realisation der reflexiven Beziehungstypen `RelationPP, RelationOO`, die auch die Beziehung zwischen Objekten zulässt, womit wir beispielsweise Objekthierarchien wie `Ort, Straße, Haus` abbilden können.

Auch wenn dieser Entwurf sehr einfach und elegant aussieht, habe ich Bedenken, wenn man dies praktisch für einen Benutzer mit einem großen Datenbestand umsetzen will. Sowohl die Anzahl der Dokumente als auch der Orte/Häuser kann recht groß werden. Der Benutzer will diese Objekte natürlich „sortenrein“ angeboten bekommen und zwar in seiner gewohnten Ordnung. Normalerweise wird er bei der Suche nach einem Haus zuerst den Ort suchen, um dann zur Straße und zum Haus zu navigieren. Da will er nicht alle Objekte, also auch die Dokumente, Bücher und Schuber sehen.

4.8.1 Datentypen der obersten Ebene (Top-Level Data Types)

Es gibt acht Datentypen der obersten Ebene (Top-Level Data Types)

Datentyp (Tabelle)	**Erklärung**
`Person`	beschreibt eine Person
`Relationship`	beschreibt die Beziehung zwischen zwei Personen
`SourceDescription`	beschreibt die Quelle mit genealogischen Informationen
`Agent`	definiert jemanden oder etwas, der genealogische Daten betreut, z. B. einen genealogischen Forscher, einen Benutzer von Software oder eine Organisation.
`Event`	beschreibt ein historisches Ereignis
`Document`	definiert das grundlegende konzeptionelle Modell für genealogische Daten, die als Textdokumente verwaltet werden.
`PlaceDescription`	beschreibt die Details eines Ortes anhand seines Namens und möglicherweise seines Typs, Zeitraums und/oder einer geografischen Beschreibung – die als Beschreibung eines Ortes als Momentaufnahme in der Zeit fungiert.
`Group`	beschreibt eine Gruppe von Personen. Das Konzept einer "Gruppe" erfasst institutionelle Assoziationen zwischen Personen, die möglicherweise direkte familiäre Beziehungen zueinander haben oder nicht. Beispiele für eine Gruppe könnten Plantagen, Waisenhäuser oder Militäreinheiten sein.

Ereignis vs. Tatsache

Ein `event` ist ein Ereignis, das zu einem bestimmten Zeitpunkt oder in einem bestimmten Zeitraum, häufig an einem bestimmten Ort, stattgefunden hat.
Ein `fact` ist ein Datenelement, von dem angenommen wird, dass es zu einem bestimmten Thema, wie einer Person oder einer Beziehung, wahr ist. Eine Zeit oder ein Ort ist oft, aber nicht immer, auf ein `fact` anwendbar. Fakten existieren nicht außerhalb des Geltungsbereichs des Themas, für das sie gelten.
Ereignisse werden oft verwendet, um auf Fakten zu schließen. Ein Heiratsereignis lässt beispielsweise darauf schließen, dass zwei Personen verheiratet waren, und ein Geburtsereignis lässt darauf schließen, dass eine Person geboren wurde (und zumindest eine Frau zur Mutter wurde). Fakten lassen manchmal auch auf Ereignisse schließen, aber die Existenz eines Fakts rechtfertigt möglicherweise nicht immer eine Beschreibung eines Ereignisses.

In diesem Manuskript werden die Begriffe (event) als Ereignis bzw. als Zustand (fact) interpretiert.

GEDCOM X gibt folgende Ereignisse vor:

URI	Ereignis
`http://gedcomx.org/Adoption`	Adoption
`http://gedcomx.org/Birth`	Geburt
`http://gedcomx.org/Burial`	Begräbnis
`http://gedcomx.org/Census`	Volkszählung
`http://gedcomx.org/Christening`	Taufe bei der Geburt, nicht jedoch eine Erwachsenentaufe
`http://gedcomx.org/Death`	Tod
`http://gedcomx.org/Divorce`	Scheidung
`http://gedcomx.org/Marriage`	Heirat

Für die Fakten sieht GEDCOM X folgende Werte vor, die jeweils noch auf einen Kontext beschränkt sind:

URI	Fakt	Bereich
`http://gedcomx.org/Adoption`	Person ist adoptiert.	`person`
`http://gedcomx.org/Birth`	Person ist geboren.	`person`
`http://gedcomx.org/Burial`	Person ist nach dem Tod beerdigt.	`person`
`http://gedcomx.org/Christening`	Person ist Kind-getauft. Eine Erwachsenentaufe ist damit nicht bestätigt.	`person`
`http://gedcomx.org/Death`	Person ist verstorben	`person`
`http://gedcomx.org/Residence`	Person ist wohnhaft.	`person`
`http://gedcomx.org/Divorce`	Paar ist geschieden.	`couple relationship`
`http://gedcomx.org/Marriage`	Paar ist verheiratet.	`couple relationship`

4.8.2 Nachschlagtabellen (Component-Level Data Types)

Weiter werden im GEDCOM X-Vorschlag auch 22 Nachschlagtabellen (Component-Level Data Types) beschrieben.

Nachschlagtabelle	Erklärung
`Identifier`	definiert die Datenstruktur, die zur Bereitstellung eines Identifizierers einer genealogischen Ressource verwendet wird.
`Attribution`	definiert die Datenstruktur, die verwendet wird, um zuzuordnen, wer, wann und warum genealogische Daten verwendet werden. Die Daten werden dem Agenten zugeordnet, der die Art der zuzuordnenden Daten zuletzt erheblich geändert hat. Die Definition einer „signifikanten Änderung“ liegt außerhalb des Umfangs dieser Spezifikation und ist dem Implementierer der Anwendung überlassen.
`Note`	definiert eine Notiz, die aus genealogischen Untersuchungen stammt.
`TextValue`	definiert einen literalen Text.
`SourceCitation`	definiert einen Container für die Metadaten, die ein Agent benötigt, um den Quelle(n).t-Wert zu identifizieren.
`SourceReference`	definiert einen Verweis auf eine Quellenbeschreibung.
`EvidenceReference`	definiert einen Verweis auf Daten, die zum Ableiten der angegebenen Instanz von `Subject` verwendet werden. Beispielsweise kann sich ein „Beweis“ `Subject` (d. h. das Objekt, das die `EvidenceReference`-Instanz enthält) auf einen Inhalt beziehen, der aus einer Quelle extrahiert wurde (d. h. ein „extrahiertes“ `Subject`), als Information, die zum Ableiten der in diesem `Subject` ausgedrückten Beweise verwendet wird.
`OnlineAccount`	definiert eine Beschreibung eines Kontos eines Onlinedienstanbieter.
`Address`	definiert eine Straße oder Postanschrift einer Person oder Organisation.
`Conclusion`	definiert das abstrakte Konzept (Schlussfolgerung) für ein grundlegendes genea-

	logisches Datenelement.
`Subject`	definiert das abstrakte Konzept eines genealogischen Betreffs.
`Gender`	definiert das Geschlecht einer Person (auch divers, unbekannt).
`Name`	definiert einen Namen einer Person (ggf. mehrfach angelegt).
`Fact`	definiert ein Datenelement, von dem angenommen wird, dass es zu einem bestimmten Thema, z. B. einer Person oder einer Beziehung, wahr ist. Informationen zum Unterscheiden des Begriffs „Fakt“ von „Ereignis“ finden Sie unter Ereignisse versus Fakten.
`EventRole`	definiert eine Rolle, die eine Person in einem Ereignis spielt.
`Date`	Datum definiert ein genealogisches Datum.
`PlaceReference`	definiert einen Verweis auf eine Ortsbeschreibung.
`NamePart`	dient zum Modellieren eines Teils eines vollständigen Namens verwendet, einschließlich der Begriffe, aus denen dieser Teil besteht. Einige Namensteile können Qualifizierer aufweisen, um dem Namensteil eine zusätzliche semantische Bedeutung zu verleihen (z. B. „Vorname“ oder „Nachname“).
`NameForm`	definiert eine Darstellung eines Namens (eine „Namensform“) innerhalb eines bestimmten kulturellen Kontexts, z. B. einer bestimmten Sprache und eines bestimmten Skripts.
`Qualifier`	definiert die Datenstruktur, die verwendet wird, um einem bestimmten Datenelement zusätzliche Details, Anmerkungen, Tags oder andere qualifizierende Daten bereitzustellen.
`Coverage`	definiert die Datenstruktur, mit der Informationen zur Abdeckung einer Ressource bereitgestellt werden.
`GroupRole`	definiert eine Rolle einer Person in einer Gruppe.

4.8.3 GEDCOM X Beziehung (relationship)

Der Vorschlag GEDCOM X verzichtet ganz auf den Datentyp `family` und ersetzt ihn durch `relationship`. Er besitzt folgende Eigenschaften

Eigenschaft	Erklärung	Beschränkungen
`type`	Aufzählungswert, der den Typ der Beziehung angibt.	OPTIONAL. Falls angegeben, MUSS ein Beziehungstyp angegeben werden.
`person1`	Verweis auf die erste Person in der Beziehung.	OLIGAT. MUSS eine vorhandene Person sein.
`person2`	Verweis auf die zweite Person in der Beziehung.	OLIGAT. MUSS eine vorhandene Person sein.
`facts`	Die Fakten über die Beziehung.	OPTIONAL.

Die Beziehungen sind zeitlos. Eine Beziehung ändert den Zustand der beteiligten Personen Lediglich in den Fakten ist ein Datum vorgesehen. Die empfohlenen Beziehungstypen

URI	Erklärung
`http://gedcomx.org/Couple`	Eine Beziehung eines Paares von Personen.
`http://gedcomx.org/ParentChild`	Eine Beziehung von einem Elter zu einem Kind.
`http://gedcomx.org/EnslavedBy`	Eine Beziehung von einer versklavten Person zum Sklavenhalter oder Sklavenhalter der Person.

überraschen uns mit dem Aspekt der Sklavenhaltung. Der Datentyp `relationship` gibt eine Reihenfolge vor. Die scheinbare Gleichberechtigung `Couple` kann erst im Programm erkannt und aufgelöst werden.

5 GEDCOM und RDBM

5.1 Normalformen

Die derzeit gängigen Datenbanksysteme beruhen auf dem Relationale Datenbankmodell (RDM), das auf fünf aufeinander aufbauenden Normalformen beruht, wobei die ersten drei technisch einfach umzusetzen sind. Die Theorie soll hier jedoch nicht weiter vertieft werden.

An einigen Stellen verstößt GEDCOM schon gegen die 1NF: Atomarität. GEDCOM erlaubt Felder (Tags), die zusammengesetzt sind: Vornamen = Rufname plus Zusatzname, Adressen usw. In den Records wimmelt es von mehrwertigen Eigenschaften.

5.2 Transformationsregeln

Genau genommen modelliert GEDCOM wegen der Rückverweise (`INDI->FAM, FAM->INDI`) ein ODBM (Objektorientiertes Datenbankmodell). Das aber technisch auf ein ORDBM (Objektrelationales Datenbankmodell) umgesetzt wird. Wir können also ohne schlechtes Gewissen eine GEDCOM-Datei gleich in eine RDB umwandeln.

Rückverweise entstehen dadurch, dass Individuen den Familien zugeordnet werden. Umgekehrt finden wir in den Familien Zeiger auf die zugehörigen Familienmitglieder. Beide Zeiger sollten „passen“, also „gültig“ sein.

Wie bereits erwähnt, verletzt GEDCOM mit dem Tag `NAME` bereits die 1NF (Erste Normalform), was wir erst einmal stillschweigend tolerieren.

Entscheidend zur Darstellung der Beziehungstypen sind die Multiplizitäten (Vielfachheiten) der einzelnen Elemente. Wir vergleichen die Begriffe und stellen folgende Transformationsregeln auf:

GEDCOM-Record	**Datensatz einer Tabelle**
`n @<XREF:xxx>@ Kennung {1:1}`	Primärschlüssel (eindeutig)
`n+1 Kennung @<XREF:yyy>@`	Fremdschlüssel
`n+x Kennung {1:1}`	Mussfeld (Attribut)
`n+x Kennung {0:1}`	Kannfeld
`n+x Kennung {0:M}`	externe Tabelle mit Rückverweisen (Fremdschlüssel)
`n+x Kennung @<XREF:yyy>@ {1:1}`	Mussverweis
`n+x Kennung @<XREF:yyy>@ {0:1}`	Kannverweis
`n+x Kennung @<XREF:yyy>@ {0:M}`	mehrwertiges Feld => externe Tabelle und Zwischentabelle
`<<structure>>`	eine Zusammenfassung mehrerer Felder, Kennungen werden einzeln aufgelöst

Beispiel:

```
FAM_RECORD :=
n @<XREF:FAM>@ FAM                                {1:1}
+1 RESN <RESTRICTION_NOTICE>                      {0:1}
+1 <<FAMILY_EVENT_STRUCTURE>>                     {0:M}
+1 HUSB @<XREF:INDI>@                             {0:1}
+1 WIFE @<XREF:INDI>@                             {0:1}
+1 CHIL @<XREF:INDI>@                             {0:M}
```

Die Familientabelle enthält diese Datensätze. Die nicht notwendigen Eltern (HUSB, WIFE) verweisen auf je eine Person XREF:INDI. Es gibt nur jeweils einen Elternteil unterschiedlichen Geschlechts. Die referenzierten Personen können daneben zu anderen Familien gehören. Es handelt sich um einen ternären, assoziativen

{0:1} Mann : {0:1} Frau : {0:M} Kinder

Beziehungstyp. Der Datensatz entsteht durch ein FAMILY_EVENT, eine Heirat im Sinne der Mormonen. Entsprechend den Regeln kann auch eine Geburt durch gleichzeitige Elternschaft einen FAM–Datensatz erzeugen.

Für die (beliebig vielen) Kinder muss im RDM eine Zwischendatei angelegt werden, die XREF:FAM und XREF:INDI als Fremdschlüssel enthält. Beide Felder zusammen sind der Primärschlüssel der Zwischendatei. Auch hier stellt sich Fragen wie:

1. Kann ein Kind auch Kind einer weiteren Familie sein? Biologisch sollte das nicht möglich sein.
2. Wie gründet ein Kind eine eigene Familie?
3. In einer Familie können beliebig viele Ereignisse auftreten. Wie unterscheiden sich diese von den Individualereignissen?

Ein Hauptproblem stellt die bereits erwähnte referenzielle Integrität dar, da möglicherweise früh referenzierte Personen erst später in der GEDCOM-Datei auftauchen, eingelesen und angelegt werden können.

Ein weiteres Problem stellen die mehrwertigen (nicht atomaren) Felder (multivalue fields) mit der Multiplizität {x:M} dar. Hier reicht ein einfacher Fremdschlüssel nicht aus, da er nur einen Wert enthalten kann. Vielmehr benötigen wir eine mx:mx-Zwischentabelle. Einige DBMS-Sprachen erlauben diese Felder (MS Access ab 2007, nicht jedoch SQL Server), indem sie intern eine versteckte Datei anlegen und verwalten. Diese Technik hat angeblich Vorteile für den DB-Einsteiger.
Insider finden dazu in Access ausgeblendete Systemtabellen wie beispielsweise MSysComplexColumns. In dieser werden die mehrwertigen Felder eingetragen und mit Systemobjekten verlinkt.

5.3 Beurteilung

Unsere Transformationsregeln sorgen für eine Fülle von Zusatztabellen, die in der Realität so gut wie nie praktisch eingesetzt werden. Denken wir nur an die erlaubten Namenswechsel, die höchst selten auftreten und vom Empfängerprogramm wahrscheinlich nicht ausgewertet werden. Teilweise werden wir die Tabellen nach dem Einlesen in eine RDM-DB verarbeiten und dann wieder auflösen.

Grundsätzlich müssen wir verschiedene Einlese-Techniken unterscheiden:

1. Ein-Pass-Technik: Die GEDCOM-Datei wird nur einmal durchgearbeitet.
2. Mehr-Pass-Technik: Die GEDCOM-Datei wird mindestens zweimal durchlaufen.

Um es praktisch an der Ein-Pass-Technik zu erklären:

1. Wir legen leere Tabellen für die in GEDCOM vorgesehenen Tabellen inklusive aller mehrwertigen Tabellen an.
2. Wir lesen die GEDCOM-Datei einmal sequentiell ein.
3. Bei jedem RECORD-Tag hängen wir einen neuen Datensatz in der entsprechenden Tabelle ein.
4. Die Werte der Folgezeilen werden in die Felder eingetragen.
5. Alle Fremdschlüssel (die nach Vorgaben eine Multiplizität (Vielfachheit, UML) {0:1} werden ohne Prüfung übernommen.
6. Alle Fremdschlüssel mit einer Multiplizität {0:M} werden erst einmal in die mehrwertigen Tabellen geschrieben.
7. Nach dem vollständigen Einlesen der GEDCOM-Datei kann eine Überprüfung starten.
   Sind beispielsweise alle Individuen vorhanden, die in den Familien eingetragen sind?
8. Sind alle Referenzen befriedigt, so können wir an eine Bereinigung denken.
   Die Rückverweise werden auf einseitige Verweise reduziert.

Bei der Mehr-Pass-Technik werden in einem Durchlauf (Pass) gezielt Datensätze erfasst, die zur Auflösung der Referenzen in einem späteren Durchlauf notwendig sind.

1. Wir legen leere Tabellen für die in GEDCOM vorgesehenen Tabellen inklusive aller mehrwertigen Tabellen an.
2. Wir lesen die GEDCOM-Datei ein erstes Mal sequentiell ein.
3. Für alle INDI-Records wird ein Datensatz angelegt. Platzhalter-Datensätze werden aufgefüllt. Unerlaubte Doppelschlüssel werden erkannt und gemeldet.
4. Für alle Referenzen @<XREF:INDI>@ auf Individuen in den anderen Records wird deren Existenz geprüft. Fehlt das Individuum noch, wird ein Platzhalter angelegt.

5. Nach dem vollständigen Einlesen der GEDCOM-Datei wird geprüft, ob es noch leere Platzhalter gibt, die ein Zeichen für eine Referenz auf ein nicht vorhandenes Individuum darstellt.
6. In den Folgedurchgängen werden bei jedem Einlesen die weiteren GEDCOM-Records erfasst und angelegt.
7. usw.

Sollen nun die GEDCOM-Daten in eine vorhandene Datenbank eingemischt werden, die vereinfachend angenommen die gerade beschriebene Struktur besitzt, so müssen alle Schlüssel (Primär- und Fremdschlüssel) umgesetzt werden, wenn sie mit vorhandenen Schlüsseln kollidieren. Erst dann kann Datenabgleich auf Duplikate erfolgen (siehe Kapitel „GEDCOM-Dateien zusammenführen“).

6 GEDCOM-Datenaustausch

6.1 Informationsverluste

Nach Definition dient das GEDCOM-Format zum Datenaustausch. Hierbei kann es an verschiedenen Stellen zum gewollten oder ungewollten Informationsverlust kommen (Abb. 7)

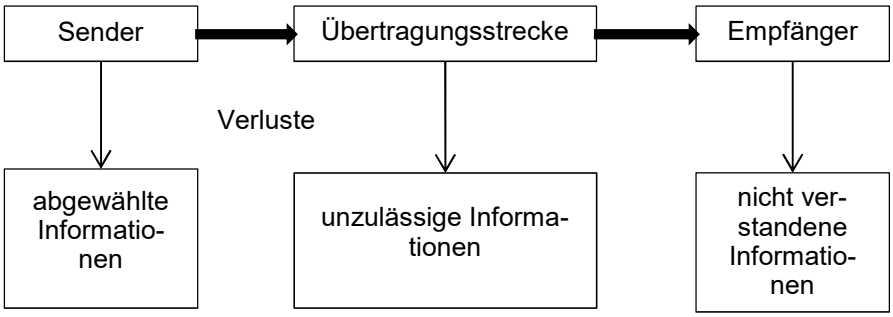


Abb. 7: Mögliche Verluste bei der Informationsübertragung über das GEDCOM-Format

Sender	Das sendende Programm erlaubt das Abwählen von Informationen. Der Benutzer schränkt die veröffentlichten Informationen selbst ein. Das Programm „erfindet“ Tags für seine Daten usw. Die Anbieter veröffentlichen selten die spezifischen Tags. Die GEDCOM-Datei enthält auch keine Liste über die abgewählten Tags bzw. Erläuterungen zu den spezifischen Tags. Leider ändert sich das Export-Verhalten auch noch von Version zu Version.
Übertragungsstrecke	GEDCOM hat keine passenden Tags für die Informationen. Der Zeichensatz ist nicht erlaubt usw. GEDCOM 5.5.5 erkennt zumindest am fehlenden `TRLR`-Tag einen Übertragungsfehler.
Empfänger	Das empfangende Programm kennt die (erfundenen oder auch erlaubten) Tags nicht. Der Importeur beschränkt die Anzahl der Tags beispielsweise bei der Anzahl der eingelesenen Namen. Der Empfänger listet die nicht verarbeiteten Informationen nicht in einer Log-Datei auf, gibt keine Korrekturhinweise, verschleiert eigene Programmschwächen. Es kommt mit dem Zeichensatz nicht zurecht usw.

An einigen Stellen in diesem Text haben wir gelernt, dass GEDCOM 5.5.1 **unvollständig**, ja sogar **fehlerhaft** ist. Deshalb ist jedes Programm, welches GEDCOM-Dateien erstellt eine *Interpretation* des Standards durch den Programmierer. Die Version 5.5.5 soll diese Fehlinterpretationen beseitigen.

6.2 Test

6.2.1 Selbsttest

Ein einfacher Test besteht erst einmal darin, auf beiden Seiten dasselbe Programm einzusetzen. Es sollte doch zumindest seine Daten wieder einlesen können. Hierbei stellt man schnell fest, dass die abgewählten Informationen definitiv weg sind. An dieser Stelle muss der Forscher entscheiden, ob er das wirklich will.

6.2.2 Fremdtest

Theoretisch müssen wir jedes Programm als Sender und Empfänger nutzen. Leider ist der Aufwand dabei mit n! (n-Fakultät) nicht zu bewältigen. Also warten wir auf Rückmeldungen der Anwender und richten eine Community-Seite ein.

6.3 Validation

Bei jeder formalen Sprache sucht man natürlich schnell nach Validatoren, die das Ergebnis nach dem Export oder vor dem Import auf formale Fehler überprüft. Hierzu finden wir

https://www.gedcom.org/validators.html

eine Liste möglicher Programme. Diese unterscheiden sich bei der Anwendung im lokalen Umfeld oder als Internet-Anwendung.

Die Auswahl ist leider enttäuschend, da viele der Programme nicht mehr gewartet werden. Versionen für 5.5.5 sind angekündigt. Für den lokalen Einsatz bleibt nur der Chronoplex GEDCOM Validator übrig.

GEDCOM 5.5.5 stellt eine Reihe von Testdateien zur Verfügung, mit denen man zumindest einen eigenen GEDCOM-reader testen kann

https://www.gedcom.org/samples.html

6.3.1 Formale Validierung

Ein GEDCOM-Validator prüft die Daten auf die formale Einhaltung der GEDCOM-Standards. Hiervon sind die logischen Validatoren zu unterscheiden, die beispielsweise Dubletten finden oder die zeitliche Plausibilität der Datumsangaben prüfen. Der Aufruf von Google bringt erstaunlich wenige Treffer. Eine Übersicht über mögliche Programme versprechen

http://genwiki.de/Kategorie:GEDCOM-Validierung

https://www.gedcom.org/validators.html

Die Programme unterscheiden sich als Anwendung auf dem lokalen PC oder als Internet-Anwendung.

Die Ergebnisse sind aber eher ernüchternd (fast leere Seiten, tote Links). Bisher sind mir nur folgende Programme zur Validierung bekannt:

http://ofb.hesmer.name/main_gsp_d.html http://genwiki.genealogy.net/ Gedcom_Validierung_(Genealogiesoftware)	GSP - Gedcom Service Programme	Strukturanalyse kostenpflichtig
https://chronoplexsoftware.com/gedcomvalidator/	GEDCOM Validator	Freeware
http://ged-inline.elasticbeanstalk.com/validate	GEDCOM Validator	Online-Validator
https://www.tamurajones.net/GEDCOMValidation.xhtml	Übersicht und Vergleich	Kein eigenes Programm

Für den lokalen Einsatz bleibt nur der Chronoplex GEDCOM Validator übrig, für den eine Version für GEDCOM 5.5.5 angekündigt ist.

6.3.2 Hilfsweise (manuelle) Validierung

Da wir GEDCOM-Dateien einigermaßen gut lesen können. Kann ein Blick auf den Inhalt mit einem geeigneten Editor die gröbsten Fehler erkennen lassen.

NotePad++ hat in der 32-Bit-Version ein Plug-In, um die GEDCOM-Tags farbig herauszuheben. Vor der Kontrolle der Struktur suchen wir aber erst einmal nach Zeichenfehlern (Abb. 8). Eine fehlerhafte Umsetzung der UTF-8-Zeichen erkennen wir daran, dass das erste Byte `hex C3` als `Ã` dargestellt wird. Die Folgezeichen auf der neuen Zeile hängen vom Original ab.

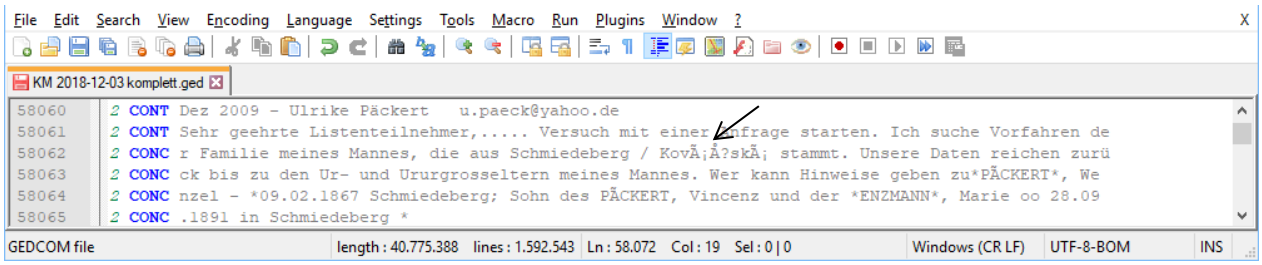


Abb. 8: Zeichensatzprobleme

Öffnen wir beispielsweise eine GEDCOM-Datei in Word, so erfolgen oft automatische Korrekturen und Änderungen, so dass nach dem erneuten Speichern „weiche Zeilenschaltungen“, Trennzeichen usw. eingefügt worden sind.

Im vorliegenden Beispiel aus PAF erscheint auch die Zuordnung von `CONT` (konkatenieren) und `CONC` (nach Zeilenschaltung verbinden) eher zufällig als sinnvoll.

6.3.3 Logische (semantische) Validierung

Die Plausibilitätsprüfung der Daten (nicht der Struktur) geht von einer fehlerfreien Struktur der GEDCOM-Datei aus.

Die Prüfungen erfolgen mit den Schwerpunkten:

- Dopplungen
- referenzielle Integrität (fehlerhafte Verweise)
- Fehler im zeitlichen Ablauf (z. B. Ereignisse außerhalb der Lebenszeit)
- Unwahrscheinlichkeiten in der Vererbung (zu jung, zu alt)
- Unwahrscheinlichkeiten in der örtlichen Verteilung

Die logische Validierung ist keine GEDCOM-Aufgabe sondern eine Aufgabe des verarbeitenden Programms. Eine Reihe von öffentlich zugänglichen Programmen führt eine solche Validierung durch.

7 GEDCOM-Dateien zusammenführen

Neben der reinen Übertragung eines Datenbestandes von einem Programm zum anderen dient GEDCOM auch dazu, Datenbestände unterschiedlicher Ahnenforscher zusammenzuführen. So entstehen mit der Zeit sehr große Datenbestände, die ein Einzelner kaum erstellen kann.

Dabei sollen möglichst alle Dopplungen erkannt und entfernt werden. Hierbei sind zwei Techniken zu erkennen:

1. Erkennung während des Zusammenführens (Problem: Zeitaufwand)
2. Nachträgliche Löschung (meist angewandt)

Das Erkennen von Dopplungen ist sehr zeitintensiv, da oft noch einmal alle Quellen untersucht werden müssen. Daher wird meist das nachträgliche Löschen eingesetzt.

7.1 Gleichheiten erkennen

Die Gleichheit des Nachnamens zweier Personen ist sicher nicht ausreichend, da wir ganze Sippen mit gleichen Nachnamen verarbeiten. Bei jedem Knoten in unserem Ahnenbaum entsteht ein weiterer Nachname (mit Ausnahme des Ahnenschwundes). Also müssen wir zusätzliche Kriterien auswerten. Theoretisch können dies alle Kriterien sein, die wir erfassen. Typisch ist der Vergleich der Vornamen sowie der Eckdaten wie Geburt, Tod usw.

Untersuchen wir Datumsangaben, so ist die Gleichheit eher unwahrscheinlich. Daher wird die Prüfung auf Intervallgleichheit zu einem besseren Ergebnis führen. Passt also ein Geburtsdatum `01.01.1899` mit einer Angabe `vor 1900` zusammen?

7.2 Ähnlichkeit erkennen

Zuerst einmal müssen wir klären, wann zwei Personen gleich oder zumindest ähnlich sind. Schon bei der ersten, scheinbar offensichtlichen Bedingung, dass die Nachnamen gleich sind, werden wir in Schwierigkeiten geraten, da die Schreibweisen je nach eingesetzter Sprache, ja je nach Schreiber unterschiedlich sein kann. Auch

haben sich die Nachnamen erst im Laufe der Zeit gefestigt. Um eine Ähnlichkeit automatisch zu berechnen, gibt es mehrere Algorithmen.

Eine Sammlung der Vergleichsalgorithmen in C# finden wir unter

https://archive.codeplex.com/?p=sounditout

mit folgender Beschreibung

```
NYSIIS (2010-11-28)
Soundex (2010-11-29)
NGrams (2010-12-04)
Caverphone (2010-12-04)
Metaphone (2010-12-05)
MatchRating (both key generation and comparison rules) (2010-12-06)
Levenshtein Distance (2010-12-07) - Not fully tested
Hamming DIstance (2011-10-24) - Not tested
Dice's Coefficient (2011-10-25) - Not tested https://en.wikipedia.org/wiki/Sørensen–Dice_coefficient
```

Einige der wichtigeren Techniken sollen kurz beleuchtet werden. Grundsätzlich können wir zwei wichtige Gruppen von Algorithmen unterscheiden.

- Die eine Gruppe liefert pro Namen einen Suchwert, der dann auf Gleichheit mit anderen Suchwerten überprüft wird.
- Eine zweite Gruppe liefert einen Ähnlichkeitskoeffizienten (Abstandsmaß) für je zwei Namen.

7.2.1 Soundex

Das klassisch Soundex-Verfahren (auch Russel-Soundex) zur Indizierung von Namen (Wörtern) für die englische Sprache wurde bereits 1918 patentiert. Dies hat den Vorteil, dass der Patentschutz inzwischen abgelaufen ist.

Aus jedem Namen wird ein Kode (Schlagwort) gebildet, der aus dem ersten Buchstaben und drei Ziffern besteht. Jede Ziffer entspricht dabei einem ähnlich klingenden Buchstaben. Nach drei Ziffern wird das Verfahren abgebrochen. Fehlende Ziffern werden auf 0 gesetzt.

Für alle Buchstaben nach dem ersten (jetzt meist kleinen Buchstaben) gelten weitere Regeln. Da die Vokale (inklusive des y) im Englischen ganz unterschiedlich ausgesprochen werden können, werden sie einfach weggelassen. Dasselbe gilt für die Dehnungsbuchstaben h und w. Die verbleibenden Konsonanten werden entsprechend der folgenden Tabelle in Ziffern umgewandelt:

Ziffer	Buchstabengruppe
1	b, f, p, v
2	c, g, j, k, q, s, x, z
3	d, t
4	l
5	m, n
6	r

Jetzt werden noch die Dopplungen entfernt und das komprimierte Schlagwort ist fertig. Mathematisch handelt es sich um die (nicht umkehrbare) Abbildung einer Menge von Namen auf eine kleinere Menge, wobei Dopplungen anzunehmen sind.

7.2.2 NYSIIS (New York State Identification and Intelligence System)

NYSIIS ist eine Variante von Soundex, das insbesondere die Anfangs- und Endbuchstabengruppen eines Namens genauer untersucht. Dadurch wird die Treffergenauigkeit um etwa 2,7 % gegenüber Soundex erhöht. Jedoch ist der Algorithmus stark auf amerikanische Namen zugeschnitten.

7.2.3 Soundex für die deutsche Sprache

Einmal davon abgesehen, dass der Anfangsbuchstabe sehr dominant ist (Katharina und Catharina ergeben abweichende Schlagwörter), unterscheiden sich der Klang der beiden Sprachen sowie die Zeichensätze (Umlaute, ß). Man setzt daher folgende, besser angepasste Buchstabenkodes ein:

Ziffer	Buchstabengruppe
0	a, e, i, o, u, ä, ö, ü, y, j, h
1	b, p, f, v, w
2	c, g, k, q, x, s, z, ß
3	d, t
4	l
5	m, n
6	r
7	ch

Indem wir die Nullen im Ergebnis entfernen, entfallen alle Umlaute, das als Vokal gesprochene Y, das Dehnungs-H und das J.

Neben den Umlauten besitzt die deutsche Sprache Buchstabenkombinationen, die besondere Laute darstellen. Einige davon werden (im Wort) automatisch umgewandelt, jedoch nicht im Anlaut

sch → 110 → 1

pf → 11 → 1

ph → 10 → 1

usw.

Das ß gibt es inzwischen zwar auch als Großbuchstaben, kommt aber nie als Anlaut vor und wird als einfaches S gewertet.

Das Rachen-Ch gibt es im Englischen nicht, während es im Deutschen in 'ich, ach, huch' eine tragende Rolle spielt.

Da deutsche Wörter im Durchschnitt länger als englische sind, können wir uns auch überlegen, die Länge des Stichwortergebnisses zu verlängern.

7.2.4 Kölner Verfahren

Das Kölner Verfahren (1969, Hans-Joachim Postel) (auch Kölner Phonetik) versucht noch stärker den Klang eines Wortes in einem künstlichen Stichwort zu berücksichtigen als es die reinen buchstabenbezogenen Soundex-Varianten tun. Dazu werden zusätzlich die benachbarten Buchstaben (Kontext) zur Bildung einer Ziffer zwischen 0 und 8 berücksichtigt.

Weiterhin gibt es besondere Regeln für den Anlaut am Wortanfang. Auch wird die Länge des Stichworts nicht beschränkt. In der folgenden Tabelle sind die deutschen Sonderzeichen eingearbeitet sowie die Regeln für c präzisiert worden.

Ziffer	Buchstabengruppe	Kontext
0	a, e, i, j, o, u, y, ä, ö, ü	
–	h	
1	b	
	p	nicht vor h
2	d, t	nicht vor c, s, z
3	f, v, w	
	p	vor h
4	g, k, q	
	c	im Anlaut vor a, h, k, l, o, q, r, u, x
		vor a, h, k, o, q, u, x außer nach s, z
48	x	nicht nach c, k, q
5	l	

Ziffer	Buchstabengruppe	Kontext
6	m, n	
7	r	
8	s, ß, z	
	c	nach s, z
		im Anlaut außer vor a, h, k, l, o, q, r, u, x
		nicht vor a, h, k, o, q, u, x
	d, t	vor c, s, z
	x	nach c, k, q

Der Algorithmus dazu lautet:

1. Wandle alle Buchstaben in klein um.
2. Entferne alle nicht erwähnten Zeichen (Bindestriche in Doppelnamen, Leerstellen in Vornamen usw.).
3. Setze die Buchstaben entsprechend der Tabelle von links nach rechts um.
4. Entferne alle Dopplungen.
5. Entferne alle Nullen ab der zweiten Stelle.

Die Erweiterung gegenüber Soundex besteht darin, dass mindestens zwei Buchstaben (Original- und Folgebuchstabe) untersucht werden.

Grundsätzlich bleibt es uns noch freigestellt, was in der Zeichenkette steht. Vornamen können so zusammen oder getrennt verschlüsselt werden.

Ein ähnliches Verfahren für Englisch ist Metaphone.

7.2.5 Methaphone

Metaphone ist ein phonetischer Algorithmus zur Indizierung von Wörtern und Phrasen nach ihrem Klang in der englischen Sprache. Da sich in Englisch die Schreibweise sehr von der Aussprache unterscheiden kann (through, thru, …) werden Zeichenketten statt Einzelzeichen unter Berücksichtigung ihrer Position im Wort untersucht.

Einige Programmier-/Skriptsprachen (PHP) enthalten standardmäßig diesen Algorithmus.

7.2.6 Silbentrennung

Hier erst einmal als Idee angedacht, wäre ein Verfahren, dass die Silbentrennung der Textverarbeitungsprogramm nutzt, um Silben zu erzeugen und diese abzuspeichern, um daraus Ähnlichkeiten abzuleiten:

Sai | bert 001003
Sei | bert 001003
Sie | bert 002003
Schu | bert 004003
Schuh | bert 004003
Schuh | mann 004005
Schu | mann 004005

Aufgrund der Phonetik würde sich beispielsweise folgende Silbenliste anbieten:

1 sai
2 sie
3 bert
4 schu
5 mann

7.2.7 Übereinstimmung

Ebenfalls aus der Textverarbeitung stammt die Möglichkeit, den Grad der Übereinstimmung zweier Zeichenketten zu bestimmen. Hierauf beruht beispielsweise die Rechtschreibprüfung, welche gleichzeitig Korrekturvorschläge macht, indem sie die Wörter mit dem jeweils höchsten Grad der Übereinstimmung als Alternativen zum

falschen Wort vorschlägt. Dabei finden wir durchaus Wörter mit einem anderen Anfangsbuchstaben, der damit nicht mehr so dominant ist.

Wird eine neue Person eingegeben, so vergleichen wir diese mit allen bereits bekannten Personen. Wird eine Übereinstimmung von 100 % erreicht, so ist die Person schon gespeichert. Bei allen anderen Personen können wir einen Prozentsatz der Übereinstimmung berechnen. Durch Festlegung eines Schwellwertes ist es dann möglich, die manuelle Prüfung auf Gleichheit zu steuern.

7.2.7.1 Levenstein-Algorithmus

Ein entsprechender Algorithmus wurde 1965 von Wladimir Lewenstein (engl. Levenshtein) entwickelt. Wir erkennen dabei, dass der Aufwand mit steigender Personenzahl und steigender Anzahl der Felder, die zum Abgleich herangezogen werden, enorm ansteigt.

Die Levenstein-Distanz (auch Editierdistanz) zwischen zwei Zeichenketten ist die minimale Anzahl von Einfüge-, Lösch- und Ersetz-Operationen, um die erste Zeichenkette in die zweite umzuwandeln. Mathematisch ist die Levenstein-Distanz eine Metrik (Abstandsmaß) auf dem Raum der Symbolsequenzen. Rätselfreunde kennen die zumindest die Einfüge-Operationen als Pyramiden-Rätsel.

Der Rechenaufwand pro Namenspaar ist von der Ordnung O(mn), also dem Produkt der Namenslängen. Der Speicherbedarf kann durch den rekursiven Hirschberg-Algorithmus (Teile und herrsche: Zerlege die Aufgabe in Teilaufgaben) linear gehalten werden.

7.2.7.2 Plagiat-Algorithmus

Ein zweiter Algorithmus ist die Suche nach den längsten gemeinsamen Zeichenketten (LCS longest common sequence) zwischen zwei Namen. Diesen Algorithmus setzen beispielweise die Plagiat-Jäger ein (jedoch mit umgekehrter Intension). Aber auch zum Vaterschaftstest lässt er sich benutzen. Hier werden die längsten DNS-Sequenzen zwischen zwei Chromosomen (Vater und Kind) bestimmt.

7.3 Zusammenführen

Nachdem wir Dopplungen/Ähnlichkeiten in der eigenen Datensammlung oder in einer zweiten Sammlung festgestellt haben, wollen wir diese auf ein einziges Objekt reduzieren. Diese Aufgabe betrifft natürlich alle Elemente unserer Datenhaltung, also nicht nur der Personen, obwohl diese meist im Mittelpunkt unserer Betrachtungen stehen. Hier können wir *Eigenschaften* einer einzelnen Person und *Beziehungen* zu anderen Personen unterscheiden. Leider treten Unterschiede auf, die erst durch eine gesonderte Recherche aufgelöst werden können.

Grundsätzlich müssen wir eine Haupt- und eine Zweitdatenbank festlegen, wobei die Zweitdatenbank nach der Zusammenführung gelöscht werden soll (oder ins Archiv wandert).

Bei einer internen Dopplung können wir von gleichartigen Datenstrukturen ausgehen. Schwieriger wird es beim Einmischen einer zweiten, externen Datenbank. Selbst wenn wir von der GEDCOM-Konformität beider Datenbanken ausgehen, sind einige Probleme absehbar. Daher müssen wir in einer bestimmten Reihenfolge vorgehen. Betrachten wir dazu noch einmal die Tabellen (GEDCOM: records). Einen groben Überblick liefert:

Record	Tag	Zeiger
INDIVIDUAL-RECORD	INDI	FAM, INDI, SOUR, SUBM, SUBN, …
FAMILY-RECORD	FAM	INDI, SOUR, SUBM
NOTE-RECORD	NOTE	
REPOSITORY-RECORD	REPO	
SOURCE-RECORD	SOUR	
SUBMITTER-RECORD	SUBM	

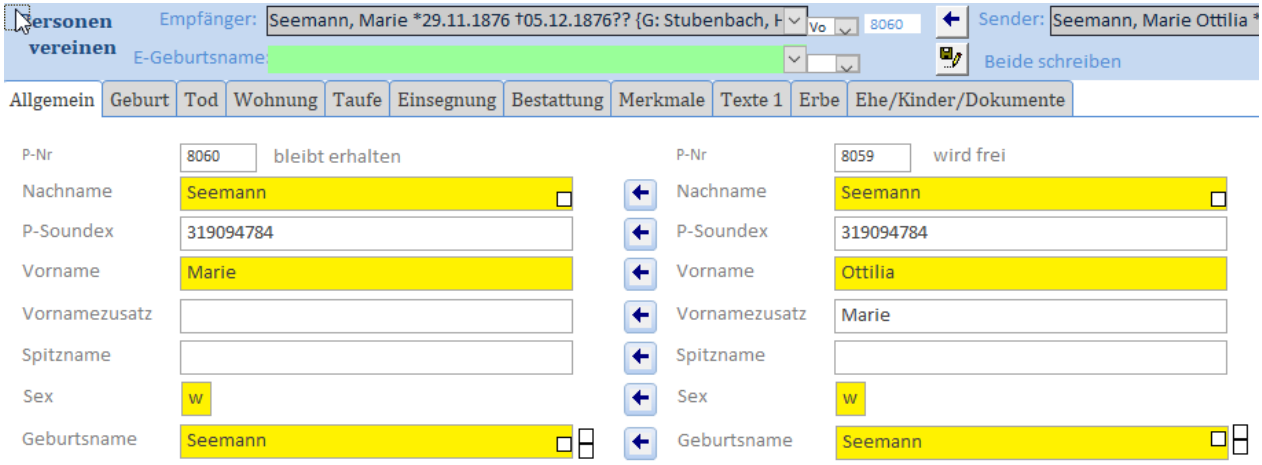
Es liegt nahe, zuerst die „zeigerlosen“ Tabellen (Mastertabellen) zusammenzuführen. Dies bedeutet in erster Linie, die Angaben zu den Quellen (Dokumenten) zu vereinheitlichen. Neben den eigentlichen Daten müssen dazu die Primärschlüssel angeglichen werden. Aber schon dabei kann es zu Konflikten kommen. Wir können nicht einfach den Wert der Hauptdatenbank `@<XREF:SOUR1>@` als Wert der Quellen in die Zweitdatenbank übernehmen, da dieser Wert dort bereits an anderer Stelle vergeben sein kann.

- Bestimme den höchsten Quellenschlüssel der Hauptdatenbank.
- Setze alle Quellenschlüssel der Zweitdatenbank auf darüber liegende Werte.
- Erkenne alle doppelten Quellenangaben und setz deren Schlüssel auf den Wert der Hauptdatenbank.
- Dies bedeutet gleichzeitig, dass ständig alle Fremdschlüssel der Zweitdatenbank nachgeführt werden müssen.
- Jetzt können wir zumindest alle Quellen oberhalb des höchsten Quellenschlüssels in unsere GEDCOM-Datenbank übernehmen.

Betrachten wir nun unsere Personen weiter.

7.3.1 Eigenschaften

Einzelne Eigenschaften einer Person sind vergleichsweise leicht zu bearbeiten, solange sie sich nicht widersprechen. In diesem Fall muss der Forscher individuell entscheiden.



7.3.2 x:mx-Beziehungstypen

Bei den x:mx-Beziehungstypen finden wir einen Fremdschlüsselwert auf der mx-Seite, der auf ein anderes Objekt wie die Familie (`FAM`), das Dokument (`SOUR`) usw. verweist.

Haupt- und Zweitdatenbank haben gleiche Quellenzeiger, um die wir uns derzeit keine Sorgen mehr machen müssen. Im GEDCOM-Datenmodell gibt es einerseits bei der Person einen Zeiger auf die Familie (mit Rollenzusatz) und andererseits einen Rückwärtszeiger bei der Familie auf die Personen.

In der Zweitdatenbank setzen wir nun zuerst die FAM-Datensatzschlüssel um

7.3.3 mx:mx-Beziehungstypen

Diese Beziehungstypen verweisen auf zwei Mastertabellen, die parallel angepasst werden müssen. Dazu müssen wir beispielsweise alle Partner durchlaufen, bereits eingetragene erkennen und neue Partner zuordnen.

8 GEDCOM und Online-Datenbanken

Fast alle allgemeinen, genealogischen Datenbanken im Internet verwenden das GEDCOM-Format, um die Informationen der einzelnen Forscher zu verarbeiten. Hier kann daher nur eine erste Einschätzung erfolgen.

Die Provider für genealogische Daten unterscheiden sich in mehreren Aspekten. Zum einen gibt es kostenpflichtige, zum anderen sind die Darstellungsformen sehr unterschiedlich. Eine Liste finden wir beispielsweise hier:

https://de.wikipedia.org/wiki/Liste_genealogischer_Datenbanken

Ein Hauptproblem stellt die Qualität der Eingabedaten dar. Grundsätzlich können wir mehrere Stufen unterscheiden:

1. Namenstabellen ohne Zeit- und Ortsangaben. Was helfen mir Stammbäume mit mehreren zehntausend Namen ohne jegliche weiteren Informationen?

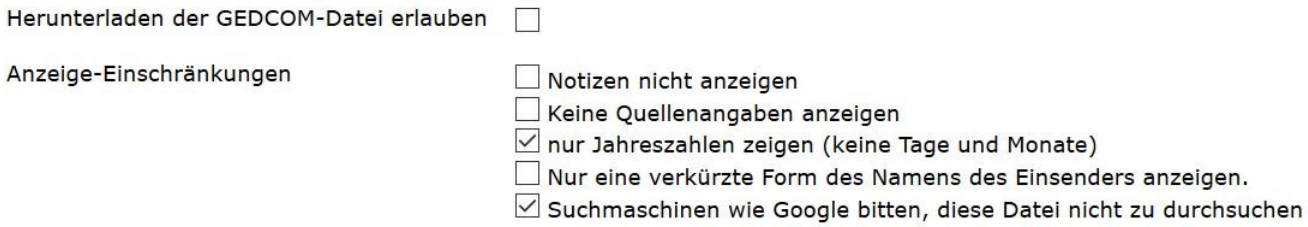
2. Namenstabellen mit Zeit- und Ortsangaben. Letztere sind aber oft nur bruchstückhaft vorhanden. Oft werden bei den (aus Geburten oder Hochzeiten) abgeleiteten Personen (Eltern) keine Jahreszahlen angegeben, so dass die zeitliche Einordnung der Personen mit gleichen Namen nicht möglich ist.
3. Namenstabellen ohne Quellenangaben. Im Fall
   „Der Einsender hat festgelegt, dass für diese Datenbank keine Quellenangaben angezeigt werden sollen."
   will der Einsender den Besucher zum „Selbstsuchen" und ggf. zur Datenkontrolle anregen. Oft aber sind die Quellenangaben verstümmelt oder verweisen auf lokale/persönliche Quellen, die nicht öffentlich zur Verfügung stehen.
4. Die genealogischen Provider arbeiten – wie GEDCOM selbst – ohne eine Ortsverwaltung, d. h. jeder Einsender kann seinen Ort nennen, wie er möchte.

Weitere Probleme ergeben sich oft, wenn man Daten zwischen verschiedenen Providern austauschen möchte, weil bei dem einen vielleicht die Eingabe einfacher ist, während der andere bessere Präsentationen besitzt.

8.1 GEDCOM und GEDBAS

8.1.1 Quellenangaben

Die wohl größte, kostenlose, deutschsprachige Datenbank ist GEDBAS, ein Projekt des Vereins für Computergenealogie e.V. Bei der Anmeldung einer eigenen Datenbank kann man einige Optionen wählen:



Je nach Gusto können nun die Häkchen gesetzt werden. Eine Diskussion über das Für und Wider zu den einzelnen Punkten würde das Thema sprengen. Vielmehr wollen wir uns der Frage zuwenden, wie können wir GEDCOM nutzen, um beispielsweise die Anzeige der Quellen zu optimieren.

Idealerweise könnte eine Quelle durch einen Klick auf einen Link die Seite eines öffentlich zugänglichen Archivs sofort aufklappen. Hier hilft ein kleiner Trick, der die Technik von GEDBAS dazu bringt, einen auslösbaren Link anzuzeigen. Dazu muss an geeigneter Stelle ein HTML-Link eingeschleust werden. Hierzu bietet sich der NOTE-Tag an:

```
1 NOTE <p># <a href="[uri]">[Anzeigetext]</a></p>
```

`[uri]`	Hier steht die Internet-Adresse der Quelle
`[Anzeigetext]`	Dieser Text wird als Quelle angezeigt.
`<p>…</p>`	Der Paragraph-Tag erzeugt eine Zeile, die mit dem # beginnt.
`<a>…</a>`	Der Anker-Tag erzeugt einen Link in einer HTML-Seite und verzweigt bei Klick nach `href`.

Das Problem besteht nur noch darin, sein lokales Stammbaumprogramm dazu zu bringen, diese Notiz zu erzeugen. Interessant sind auch immer wieder die Kommentare der Einreicher zu den Quellen:

https://gedbas.genealogy.net/person/show/1224141457

8.1.2 Ortsangaben

In GEDBAS ist jede beliebige Ortsangabe erlaubt. Nach diesen Angaben kann man auch suchen.

Filtern wir die Ergebnisse nach dem Ort, wird genau die Schreibweise des Einsenders benutzt, um eine Treffertabelle für den Ort anzuzeigen.

8.1.3 Nachname/Geburtsname/Suchname

Dieses grundlegende Problem schlägt sich natürlich auch in GEDCOM nieder. Es tauchen in den Foren immer wieder Diskussionen über die Erfassung von Bezeichnungen, insbesondere der Namen:

Geburtsname	Wir haben schon gesehen, dass dieser Name keine Invariante darstellt, da er sich durch Anerkennung der Vaterschaft auch nach längerer Zeit (teilweise Jahre) noch verändern konnte.
Nachname	Dies ist ein temporärer Name, der i. A. durch Heirat festgelegt wurde. Auch er ist nicht invariant, da er nach dem Tod des Namensgebers durch erneute Heirat oder durch ‚Annahme des Mädchennamens' verändert werden konnte. Wir sollten aber beachten, dass die Nachname beispielsweise in den Sterbebüchern ohne Angabe des Geburtsnamens auftritt. Auch sind oft den späteren Generationen nur der Nachname in Erinnerung geblieben. In den älteren Kirchenbüchern haben die Frauen oft keine Geburtsnamen.
Suchname	Der Genealoge setzt alles daran, um Dopplungen zu vermeiden. Auch sollte die Suche nach einer Person möglichst schnell, eindeutig und trotzdem trennscharf sein. Soll der Sucher alle Schreibweisen ausprobieren, oder setzen wir Schreibvarianten automatisch um? Was ist mit den Diakritika? Sollen die latinisierten (Vor-)Namen oder die typischen gespeichert werden?

In GEDBAS lernen wir, wie andere Forscher mit diesen Problemen umgehen.

Doppelte Namen:

https://gedbas.genealogy.net/person/show/1129704043

In dieser Datenbank werden zwei Namen aufgeführt, Geburts- und Nachname. Hier stellt sich die Frage, wie das in GEDCOM vorgesehen ist. Der Nachname `SURN` hat die Multiplizität {0:1}, sollte also nicht mehrfach auftreten. Möglicherweise wird eine der anderen Elemente „zweckentfremdet".

Fehlende Namen:

Hier gibt es fast unendlich viele Varianten. Angefangen von `NN` in verschiedenen Schreibweisen bis hin zu Sonderzeichen wie `…, ?, ??` usw.

Ist der Geburtsname unbekannt, so wird gern der Nachname in Klammern angegeben.

Groß-/Kleinschreibung

Einige Forscher signalisieren uns mit der Schreibweise des Nachnamens selbstdefinierte Eigenschaften. In GEDBAS sind auf klein-/groß geschriebene Nachnamen erlaubt.

GEDBAS selbst verändert die Schreibweise. So ist der Nachname auf der Suchseite normal geschrieben, taucht dann aber auf der Detailseite in Großbuchstaben auf.

Ich bin sicher, dass Sie noch viele weitere Beispiele benennen können.

8.1.4 Vorname/Rufname und Zusätze

Oft haben Personen mehrere Vornamen, deren Anzahl in Deutschland schon Gerichte beschäftigt hat. Im Laufe der Zeit haben sich die Reihenfolge und die Markierung immer wieder einmal geändert.

Meine Vornamen sind „Paul Hans-Jürgen", wobei ‚Paul' der Patenname vorangestellt ist. In Dokumenten nach 1945 wurde der Rufname „Hans-Jürgen" unterstrichen.

Die Amerikaner lieben einen Mittelbuchstaben wie „John F. Kennedy", der den Nachnamen der Mutter „Fitzgerald" abkürzt. Aber auch in Deutschland ist der Mittelbuchstabe ein Zeichen besonderer Modernität, egal was er bedeutet.

Für den Genealogen stellen mehrere Vornamen ein Problem dar, da oft der Rufname erst durch weitere Dokumente bestätigt werden kann. Logischerweise wird die Suche erheblich erschwert.

In GEDBAS wird der Suchname immer vorangestellt. Alle weiteren Vornamen werden dahinter ergänzt. Ist also der Vorname mehrteilig, so muss man alle Teile jeweils voranstellen und alle Varianten durchprobieren.

8.1.5 Sortierung

GEDCOM setzt zur Darstellung der Ereignisse Tabellen ein, die wir nach unseren Wünschen sortieren können. Dabei finden wir eine intelligente Lösung bei der Sortierung der Datumsangaben:

- Liegen für alle Ereignisse exakte Daten vor, dann ist die Sortierung in der Spalte `Art` gesperrt und in der Spalte `Datum` erlaubt. Die Sortierung erfolgt nach den Datumsangaben.
- Liegt in einer der Zeilen kein exaktes Datum vor (ungenaues Datum, fehlendes Datum), dann ist es genau umgekehrt.

9 Auswertungen

Es schließen sich jetzt noch einige Gedanken zu den Auswertungen der genealogischen Daten (im Sinne von Big Data) an, die nicht unbedingt mit GEDCOM direkt zu tun haben. Wir finden aber GEDCOM-Dateien mit mehreren hunderttausend Zeilen, die wir u. U. in eine Data-Pipeline zu weiteren Auswertungen schicken wollen. Typische Fragen sind beispielsweise:

- Altersstrukturen
- Raumstrukturen

9.1 Data Pipeline

Unter einer Data Pipeline verstehen wir den mehrstufigen Prozess, mit der unsere genalogischen Daten weiter verarbeitet werden, um sie mit einem KI-Algorithmus zu bearbeiten. Grob können wir folgende Arbeitsschritte (Workflow) unterscheiden:

- Daten einlesen
- Daten verarbeiten
- (aggregierte) Ergebnisse darstellen

Das Konzept ist keine neue Erfindung, da dies bereits über Jahre bei der Datenbank-Entwicklung umgesetzt wurde. Eine Erweiterung findet es hauptsächlich in der Technik, wie die angelieferten Daten angeliefert werden und in welchem Zustand (mit oder ohne Schema/Struktur) sie sich befinden.

Eine wesentliche Aufgabe besteht nun darin, in einer aufwendigen Vorverarbeitung die Daten so zu bereinigen, dass sie anschließend in einem einheitlichen Verfahren weiter verarbeitet werden kann. Insbesondere sollen unterschiedliche KI-Werkzeuge zur Auswertung eingesetzt werden.

GEDCOM-Dateien haben ein Schema (wie bislang ausführlich beschrieben). Letztendlich besteht unsere Aufgabe darin, die unterschiedlichsten Quellen (Kirchenbücher, Dokumente, Grabsteine, Erzählungen, usw.) entsprechend dieses Schemas in maschinenlesbaren Kode umzuwandeln und zu strukturieren (GEDCOM-Blöcke). Nachdem wir den mühsamen Weg des Transkribierens schon hinter uns haben, können wir folgende Datenquellen unterscheiden:

Dateien

Genau genommen müssen wir noch einmal eine Unterscheidung in unstrukturierte Textdateien (Geschichten, Erzählungen in Freitext), spaltenorientierte Textdateien (wie Log-Dateien) oder blockorientierte Textdateien (GEDCOM-Dateien) unterscheiden.

Sobald mehrere Personen die Daten „originalgetreu“ transkribieren, stellen wir schnell Eigenheiten der Transkribierer fest. So wird aus dem Original „ResGefr“ in einer Verlustliste des WK I schnell ein „Reserve Gefreiter“. Aber auch schon die Originale können sich in vielfacher Weise bei der Beschreibung eines Objekts unterscheiden. So ändern sich die Namen von Personen (genauer von Familien) und/oder Orten durchaus innerhalb ein und desselben Kirchenbuchs, weil der Kirchenbuchschreiber nach „Gehör“ notiert hat oder einer anderen Sprachfamilie angehörte.

Besonders anfällig für unterschiedliche Formatierungen ist das Datum. In „modernen“ Standesamtsbüchern wird die Geburtszeit vermerkt, in Taufbüchern dagegen nicht. Wird nun nur das Taufdatum erfasst, so steht die Uhrzeit auf 00:00:00, also vor der Geburtszeit, was natürlich korrigiert werden muss.

Datenbanken

Durch das Einlesen der GEDCOM-Datei werden die genalogischen Daten in eine (zumindest eingebettete, in memory) Datenbank übergeführt. Oft kann diese Datenbank für die dauerhafte Speicherung exportiert werden.

Problem: Datenbanken „zementieren“ die Betriebsabläufe einer Anwendung. Datenbankmodelle lassen sich nicht einfach von heute auf morgen umstellen. Datenbanken sind das Betriebsgeheimnis der Programmanbieter.

Daten-Seen (Data Lake)

Sehr große (operative und ggf. verteilte) Datenbestände lassen sich nicht jedes Mal für eine Anfrage vollständig durchsuchen. Außerdem belasten die Durchsuchungen oft sehr stark den operativen Betrieb. Daher werden für die Auswertung verteilte Zwischenzustände gebildet, dauerhaft vorgehalten und nur hin und wieder aktualisiert. Diese Bestände sind dann unabhängig vom operationellen Betrieb.

Serverlose Daten (Serverless Data)

Noch in der Forschung werden Programme in Containern gekapselt und als Service angeboten, so dass man den Server „dahinter nicht mehr sieht“. Sie stellen über ihre Schnittstellen damit Datenverbraucher und Datenlieferanten dar.

Grundidee dabei ist, die Daten mit verschiedenen Modellen (Algorithmen) auszuwerten, um damit beispielsweise die Kundenbedürfnisse besser zu verstehen und zu bedienen.

10 Realisation in Gramps

Bevor man eigene Entwürfe macht und umsetzt, schaut man sich natürlich nach „Konkurrenzprodukten“ um. Viele der Programme machen aus ihren Innereien ein gut gehütetes Betriebsgeheimnis. Also fällt unser Blick auf eine öffentliche, kostenlose Entwicklung namens Gramps

gramps-project.org

Anzumerken ist, dass es eine Reihe von Anwendern gibt, die Gramps für zu kompliziert halten. Trotzdem oder gerade deswegen wollen wir diese Realisation näher untersuchen. Vielleicht können wir den Grund für den Wunsch vieler Forscher nach einem „ganz einfachen Programm“ verstehen.

10.1 Datenmodell

10.1.1 Tabellen

Gramps benutzt 42 Tabellen

markertype	person	eventref
nametype	family	repositoryref
attributetype	source	personref
urltype	event	childref
childreftype	repository	mediaref
repositorytype	place	person_names
eventtype	media	person_families
familyreltype	note	person_parent_families
sourcemediatype	name	person_addresses
eventroletype	lds	repository_addresses
notetype	markup	place_locations
gendertype	address	address_locations
ldstype	location	
ldsstatus	noteref	
config	sourceref	

Die Tabellennamen stehen im Singular. Alle einfachen m:n-Zwischentabellen erkennen wir an Unterstrichen im Namen. Hier tauchen auch Plurale auf.

Alle `xxxtype`-Tabellen sind Nachschlagetabellen, also Tabellen mit Schlüssel-Werte-Paaren, die jedoch einheitlich mit einem Stellvertreterschlüssel (Surrogatschlüssel, surrogate key) zur internen Optimierung versehen sind.

Alle `xxxref`-Tabellen sind assoziative Tabellen, also m:n-Zwischentabellen mit zusätzlichen Eigenschaftsfeldern.

Neben den programmspezifischen Tabellen erkennen wir alle GEDCOM-Tabellen. Auffällig sind weitere Basistabellen wie `event, place,` die wie die GEDCOM-Tabellen eine besondere `gramps_id` besitzen.

10 Realisation in Gramps

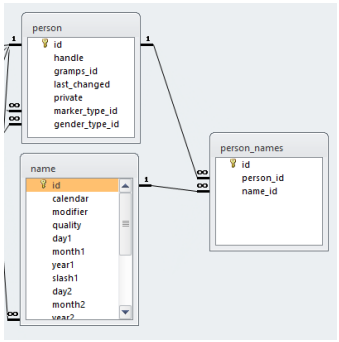


Abb. 9: Gramps-Ausschnitt

Merkwürdig ist die Einführung der Tabelle `name`, die alle persönlichen Eigenschaften einer Person (wie Vornamen, Geburts-/Todesdatum usw.) enthält und über die Zwischentabelle `person_names` mit der Tabelle `person` verknüpft ist. Das bedeutet, dass eine Person mehrere Namen inklusive mehrere Datumsangaben besitzen kann und umgekehrt ein Name mit den erwähnten Details zu verschiedenen Personen gehören kann.