Core Data Model for Managing Taxonomic Names, Concepts, and associated References

Richard L. Pyle

DRAFT

Last Updated: 19 November 2002

OVERVIEW

Included below is a description of several "core" elements of the data model I have developed for managing information relevant to Taxon Names, Taxon Concepts, and References. My complete data model is much more expansive than this, but this document is limited in scope to only the most fundamental and central components associated with tracking taxon names, concepts, and associated references.

I've broken the data tables into three sections: **Taxa** (taxon names, concepts, and associated data), **Agents** (people and organizations), and **References** (publications and other forms of date-stamped information conveyance by one or more Agents). Each of these three sections includes a text-based narrative explaining the tables and fields (entities & attributes), plus a diagram or series of diagrams to illustrate the nature of the relationships among them.

The narrative includes a brief "Overview" (abstracting the overall data model subset) followed by a more elaborated description of each major table, with descriptions of individual fields. A "Limitations" section describes some of the known limitations of the relevant portion of the illustrated model.

Notation on the diagrams more or less follows that used in FGDC_TaxNom.doc file, distributed by Bob Peet as part of the summary of the FGDC Biological Data Working Group's "Biological Nomenclature/Taxonomy Meeting Summary", held in Washington DC in November 2000. I've embedded a copy of the "Legend.pdf" file from that meeting, which can be viewed by double-clicking the icon below (assuming you have Adobe Acrobat Reader installed).

Legend.pdf

For further clarification of the diagrams, within each box the table name appears at the top of the box, and within each box there are as many as four kinds of attributes:

- **Unique Keys.** The fields in the this section of each entity box represent the uniquely-identifying key fields of each table. All tables have a surrogate Primary Key, which usually takes the name of the table (minus the "tbl_" prefix), with the addition of an "ID" suffix (indicated in **bold** in the diagrams). In my implementation, these surrogate keys are almost always long integers, with automatically-assigned random/arbitrary values, with no inherent information content. In addition to the surrogate keys, if each instance (record) of a table can also be uniquely identified by one non-surrogate field, or a combination of two or more other non-surrogate fields (i.e., multi-part key), then I list these fields in this same section of each box.
- **Foreign Keys.** These are foreign key fields that serve as the link to the surrogate Primary Key of another table, but which do not constitute a portion of a multi-part unique key. They are indicated in **Red Bold** text.
- **Non-Key Attributes.** These are actual data-bearing fields, not representing foreign keys to other tables.
- **Cheat Attributes.** These are "artificial" system-level fields created solely for the purpose of enhancing multi-record processing performance. They are non-data-bearing in the sense that

they only contain derived data (i.e., derived from other fields in the parent table or in linked tables). These fields can be completely eliminated from the model without resulting in any loss in information content. Users **never** have editing access to these fields – they are maintained entirely by software code, and are only exposed to users indirectly to enhance the performance of query/search/sort activities. For the most part, these fields can be ignored for all discussions related to information modeling.

Whenever possible, in the diagrams I've drawn the lines establishing relationships among tables in such a way that the lines connect directly to the fields that participate in the relationship. One consistent exception to this is for recursive (self) relationships, where I generally align the connection point for the "many" side of the relationship to the appropriate field, but the "one" side connects to the top of the entity box, with the implication that it joins to the surrogate Primary Key. In cases where I was unable to align the lines with associated fields, the fields involved with the relationships are usually evident. For the fields with only a few defined domain values, those values are usually listed in blue text beneath or adjacent to the corresponding table box. If such lists include "etc…" at the bottom, then the list is intended to represent example values only, rather than a predefined set of values. Other comments (e.g. business rules) are added for various relationships, to enhance clarity.

I've deliberately kept this as simple as possible, both for ease of understanding, and because of my philosophy that discussion should begin with the very basic core essentials of a data model, and then build upon that core for more elaborate and robust data elements.

TAXA

Overview

There is a well-acknowledged subtle but important distinction between a "Taxon Name" and a "Taxon Concept" (=Circumscription). A "Taxon Name" is made available according to Codes of nomenclature (ICZN, ICBN, ICNB), and is generally anchored to biological entities via a primary type specimen. Attributes about that Name (publication date, spelling, authorship, etc.) are usually unambiguous, and objectively discernable. Taxon Names can be thought of as the individual "words" comprising the dictionary of the diversity of life.

A "Taxon Concept", on the other hand, is a much less discretely defined entity, the creation or establishment of which is not governed by any internationally-accepted codes, and whose attributes are considerably more ambiguous than those of a taxon name. Whereas a Taxon Name is generally anchored to the biological world via a single specimen, a Taxon Concept is intended to circumscribe a large (potentially vast) collection of individual organisms, living, dead, and yet-to-be-born, all of which share a level of common ancestry (kinship) and morphological/genetic similarity so as to be regarded as belonging to the same taxon (e.g., species). Taxon Concepts can be thought of as the definitions of those Taxon-Name "words" that comprise the dictionary of the diversity of life.

Unlike the definitions of most words in a conventional dictionary, however, the mapping of Taxon Concepts to Taxon Names has been far from consistent among practitioners of taxonomy. Some taxonomists tend to prefer more generalized concepts (=definitions), which leads to more of the names (=words) being synonymous with other names (=words). Others prefer more specific concepts (=definitions), thereby maintaining distinctions between different names (=words). The basic problem is that most published and unpublished documentation about taxa uses only the names (words), without necessarily including explicit details about how those names are circumscribed (defined). Thus, the task at hand is to find a way to consistently and objectively map names (words) to their various respective implied circumscriptions (definitions).

In order to map the names to the circumscriptions, the first step is to apply an unambiguous "handle" on each, and then build an index to map the name handles to the circumscription handles. The easiest and most straightforward way to put a "handle" on a taxon name, is to attach that handle to the Basionym of that name. Although the word "Basionym" is more frequently used in botanical contexts than in zoological contexts, the basic concept applies equally to both. The Basionym can be thought of as a pointer to a name's original description – the moment when a string of text characters becomes legitimately available for use (in accordance with the various codes of nomenclature) – and therefore as the "handle" to a name. After much contemplation, I decided to substitute the word "Protonym" in place of Basionym for my data model – primarily to avoid confusion and misconceptions that may arise due to the excess "baggage" of meaning and interpretation of the word "Basionym". Thus, when I use the term "Protonym" in this model, I mean it mostly in the same sense as a "Basionym". In any case, the Protonym/Basionym traditionally takes the form of:

"Name OriginalAuthor(s), OriginalYear".

As described in detail within the "REFERENCES" section of this document, I define a "Reference" generally as a "date-stamped Author(s)", which can also be read as "Author(s), Year". Thus, a Protonym can be thought of as:

"Name OriginalReference"

Applying a "handle" to a Taxon Concept is less universally identifiable, and not often as unambiguous as applying a "handle" on a Name. However, a common way that is used for this purpose is to cite a name in the context of another Reference, in the form of:

"Name OriginalReference *sensu* OtherReference"

Or, in the case of the circumscription associated with the Protonym itself, as:

"Name OriginalReference *sensu* OriginalReference"

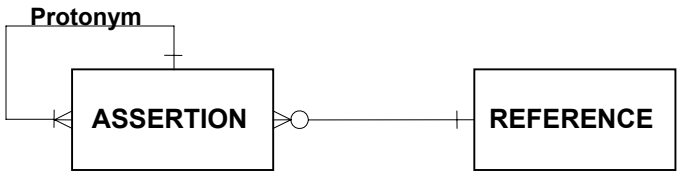
Reducing this one step further, the circumscription can be thought of as:

"Protonym *sensu* Reference"

(where "Reference" is either "OriginalReference" in the case of the circumscription attached to the original name creation, or "OtherReference" in all other cases).[1] Thus, whereas the "handle" for a Taxon Name can be thought of as the Protonym, the "handle" for a Taxon Concept can be thought of as the intersection of a Protonym and a Reference.

I have used the term "Assertion" to represent this Protonym(Name)-Reference intersection, which has previously been diagramed (e.g., in the FGDC Biological Data Working Group's "Biological Nomenclature/Taxonomy Meeting Summary") as follows:

This diagram implies a "One to ***Zero***-to-Many" relationship between names and assertions. However, a name cannot exist without at least one Assertion (at minimum, the Assertion that produced the Protonym for that name); therefore, the relationship between names and Assertions should be "One to ***One***-to-Many". Taking this one step further, given that a name cannot exist without its Protonym, and that a Protonym exists in the context of the Reference that originally established it, a Protonym can itself be represented as an Assertion. In other words, the relationship between a "Name" (Protonym) and an Assertion becomes recursive:



[1] There is an added layer of complexity in the botanical tradition of conferring special status (and authorship recognition) of those particular circumscriptions representing first use of new combinations of generic and specific names, and this is discussed in more detail below. The important point here is that the original Basionym and its authorship are retained even in the context of subsequent new combinations, such that from a logical perspective,

Therefore, the conceptual "handle" for the name and the "handle" for the concept are one and the same, with the former being a special-case subtype of the latter.

It is worth clarifying at this point that, although the "handle" to a taxon concept can be thought of as an instance of an Assertion; not all Assertions necessarily represent implied Taxon Concepts. For example, one form of publication is a "Type Catalog", wherein all type specimens in a Museum's collection are listed according to the names that they typify. In such publications, the authors will list taxon names (generally as unaltered Protonyms in this case), and hence establish an intersection between a Reference (the type catalog publication itself) and a Protonym – but without necessarily implying a Taxon Circumscription to go along with that name (i.e., literally only the "Type-Anchor" is asserted in such cases, without any implications about the scope of non-type individual kin organisms to be included within the taxon concept represented by the name). In such cases, an instance of an Assertion exists without an implied taxon concept. For this reason, an Assertion should be regarded as a "Potential Taxon [Concept]" (*sensu* Berendsohn, 1995). In the majority of Name-Reference intersections (Assertions), however, the author(s) of the Reference had a taxon concept circumscription in mind when invoking the Taxon Name, even if the scope of that circumscription is not defined (or even alluded to) within the Reference itself. Thus, in the vast majority of cases, Assertion instances can be used as a direct "handle" to an implied taxon concept circumscription (which, in many cases, will be precisely identical to the circumscriptions implied by many other Assertions for a given Taxon Name). Because my definition of a "Reference" is not restricted to publications, it can be said that all circumscriptions that map to taxon names can be identified by an Assertion.

Before describing the "Taxa" data model in detail, I want to outline what I see as alternative distinct "resolutions" at which circumscription scopes are often defined:

Name-Resolution Circumscription Definitions

This is the coarsest, and most often-used resolution of circumscription scope expression in published taxonomic references. Such circumscriptions are defined merely by treating taxon names as either valid, or as junior synonyms of other taxon names. Because taxon names are anchored to the biological world via type specimens, this method of defining circumscriptions can be thought of in a sense as Specimen-resolution circumscription definitions, except limiting it to only those particular specimens that represent primary types of taxon names. To list taxon name 'B' as a junior synonym of taxon name 'A', is to assert that "the primary type specimen of taxon name 'A' and the primary type specimen of taxon name 'B' share close enough kinship to each other that they should be regarded as belonging to the same taxon circumscription" (in this case, with the relevant Code bestowing the name 'A' with nomenclatural priority over the name 'B'). Conversely, to list taxon name 'B' as valid and distinct from taxon name 'A', is to assert that "the primary type specimen of taxon name 'A' and the primary type specimen of taxon name 'B' are sufficiently distant in kinship to each other that they should be regarded as belonging to different taxon circumscriptions". In this way, the full scope of the implied circumscription is represented by the set of Assertions within a Reference that include a name that is treated as valid, plus all assertions of names that are treated as junior synonyms of that valid name (the "handle" on the assertion being maintained as the one represented by the valid name).

---

circumscriptions can ultimately be represented as "Basionym *sensu* Reference". See the elaborated discussion in the "Limitations" section below.

The primary weaknesses of this form of circumscription definitions are as follows:

1) When a reference does not treat all relevant names that are available at the time the Reference is established (e.g., when not all potentially valid taxa are treated, or not all potentially relevant synonyms are assigned to names that are treated as valid), then the circumscription definitions within the context of the reference are incomplete.
2) Even when a reference does treat all relevant names available at the time the Reference is established, the reference may be later rendered incomplete by subsequent descriptions of new relevant names (either by splitting existing circumscriptions, or discovering new populations warranting new taxon names).
3) Using only name-level circumscription definitions (i.e., without elaborating the character-based criteria used to delineate different circumscriptions), greatly inhibits the ability to secondarily assign individual non-type specimens to these circumscriptions.

These weaknesses notwithstanding, name resolution circumscription definitions represent the bulk of documented taxonomic information, and therefore serve as an ideal "core" information content base around which the foundation of a data model should be built.

Specimen-Resolution Circumscription Definitions

The most fundamental (and finest) resolution at which circumscriptions are mapped is via individual specimens (beyond the limited scope of primary type specimens). The source Reference for corresponding Assertions can either be in the form of a publication (as when a published reference lists Museum specimen catalog numbers under a particular taxon name), or in the form of an unpublished "Determination"-type reference (i.e., identification labels on the actual museum specimens themselves).

Other Circumscription Definition Resolutions

It could be argued that "Character-Resolution Circumscription Definitions" represent a another resolution at which circumscriptions can be defined. For reasons not elaborated herein, I see this as a fundamentally different approach to mapping the scope of taxon circumscriptions, because it transcends the individual organism (considered to be the basic unit of a taxon). While this question is certainly ripe for discussion, it goes beyond the intended scope of this document. Also, circumscriptions are sometimes defined in terms of populations of organisms. This resolution of circumscription definition represents cases where a reference ascribes specific populations to taxon names, thereby extending the resolution of circumscription boundary delineation beyond the relatively course type-specimen anchor points, but not as precise as specimen-resolution definitions. This kind of circumscription definition usually takes the form of biogeographic treatments (i.e., maping taxon names directly to geographic regions, bypassing the more fundamental connection between names and locations via specimens). Although I have given some thought to modeling these sorts of circumscriptions, those thoughts have not extended much beyond preliminary ideas, and I therefore have not included any descriptions herein.

The core "Taxa" data model represented here is intended to directly document "Name-Resolution" circumscription definitions, while also providing a tangible "handle" to a

circumscription (i.e., an Assertion instance) that can be more precisely defined at higher resolution (e.g., specimen resolution) via additional "layers" of data entities.

tbl_Assertion

The central "anchor" entity of the taxon portion of this data model is the Assertion. As previously stated, an Assertion is defined as the intersection of a Protonym and a Reference, as indicated by the Foreign Keys, *ProtonymID* and *ReferenceID*. Because Protonyms themselves represent Assertions (*sensu* the original authors of the Protonym), it would be possible to represent the linkage of *ProtonymID* to *AssertionID* via a direct recursive link. However, because certain attributes apply only to Protonyms and not all Assertions (e.g., nomenclatural attributes such as *Availability* in the case of names governed by Codes, and "Type Species" and *Gender* in the case of generic-level names – describe in more detail along with other Protonym attributes below), and also for reasons of enforcing business rules and improving performance of certain query operations, I have defined the table **tbl_Protonym** as a subtype of **tbl_Assertion**. The recursive linkage between any particular Assertion instance and its associated Protonym is made via the **tbl_Protonym** subtype; first from the *ProtonymID* Foreign Key field of **tbl_Assertion** to the *ProtonymID* Primary Key of the subtype **tbl_Protonym**, and then back to **tbl_Assertion** table via the One-to-One subtype link to *AssertionID*. The domain of Assertion instances that are to be represented by instances in **tbl_Protonym** are, by definition, those instances of **tbl_Assertion** where *AssertionID=ProtonymID*.

The *ReferenceID* Foreign Key of **tbl_Assertion** is a straightforward linkage back to the Reference in which the assertion is made. Special care must be taken when establishing this link for Assertions that represent Protonyms, because the authorship of the Protonym is derived directly from the authorship of the corresponding Reference. In cases where taxon authorship does not exactly match original Reference authorship, then a new Reference of type "Sub-Reference" must be created to represent only that portion of the parent Reference constituting the description of the relevant Protonym, so that correct Protonym authorship (including cases of "ex" authors) can be ascribed to the Protonym via the Sub-Reference authors.

In the vast majority of cases, the two Foreign Keys *ProtonymID* and *ReferenceID* would (by themselves) uniquely identify every circumscription Assertion. However, in the special case of Nominotypical taxon circumscriptions (e.g., the subfamily Chaetodontinae within the family Chaetodontidae; or the subgenus *Chaetodon* (*Chaetodon*), or the subspecies *Chaetodon unimaculatus unimaculatus*) represent cases where a single Protonym can be used within a single Reference as representing two distinct taxon concept circumscriptions. For this reason, the *TaxonRankID* Foreign Key, which identifies the exact taxonomic rank at which the Protonym is used within the Reference, must also be included among the uniquely-identifying attributes of a particular Assertion instance.

The *TaxonRankID* Foreign Key establishes a link to the **tbl_TaxonRank** table. Although the final field structure of this table has not been completely identified as yet in my model, its primary element is the *RankName* (e.g., "Kingdom", "Family", "Species", "Variety", etc.)[2]. The

[2] Unfortunately, the RankNames are not universally established for all of biology. The most persistent inconsistency is the *RankName* "Phylum" in zoological nomenclature corresponding to the *RankName* "Division" in botanical nomenclature ("Division" is often used for a different rank – between Class and Order – within Zoological nomenclature). There are several solutions to this inconsistency, but I haven't yet decided which solution I prefer – which is why the final field structure is not yet completely identified.

remaining fields are not strictly core attributes of each Taxon Rank, but are used by the application for formatting purposes (*Abbreviation* is a 3-character abbreviation of each rank used as a delimiter within the *CheatHierarchy* field of **tbl_Protonym**, and *Prefix* and *Suffix* are used to format the *CheatTaxonName* field of **tbl_Assertion**). It is worth noting that the Primary Key of **tbl_TaxonRank** (*TaxonRankID*) is information-bearing in that the numeric values are assigned in sequence from the highest taxonomic rank (=lowest ID number value), to lowest taxonomic rank (=highest ID number value). To maintain consistency in non-information-bearing Primary Keys of other entities in this schema, it might be appropriate to create a new attribute for **tbl_TaxonRank** (e.g., *Sequence*) to store rank sequence information.

Although *TaxonRankID* technically serves as a component of the unique identifier for each Assertion record, it only serves a function in this capacity for those relatively few cases involving Nominotypical taxa. In a broader sense, I regard *TaxonRankID* as one of the four "basic elements" that make up the essence of an Assertion. It is necessary even outside the context of Nominotypical taxa because the same taxon name may be used to represent different taxonomic ranks (e.g., as a family or a subfamily; as a genus or a subgenus; as a species, a subspecies, a variety, or a form; etc.). Thus, to adequately describe how a taxon name was used within the context of a particular Reference, it is necessary to document exactly what rank the name was used to represent.

The second of four basic elements of an Assertion is its "Validity" (i.e., whether or not the name was treated by the Reference as a valid taxon, or as a junior synonym of another taxon). This element is documented via the *ValidAssertionID* Foreign Key, which recursively links back to the same or another instance of the **tbl_Assertion** table. All Assertion instances must indicate a value for *ValidAssertionID*. Cases where the Reference treated the name as a valid taxon are indicated by *ValidAssertionID=AssertionID* (almost by definition, this includes all Assertions that are included in the **tbl_Protonym** subtype) . In cases where the Reference treated the name as a junior synonym of another name, then *ValidAssertionID≠AssertionID* for the given instance, and *ValidAssertionID* instead points to the Assertion that represents the indicated senior synonym of the original instance.

At the moment, I am imposing the restriction that when *ValidAssertionID≠AssertionID*, the *ValidAssertionID* must point to a different Assertion instance based on the exact same *ReferenceID* as the original Assertion instance. In other words, inter-Assertion linkages via this Foreign Key must be established *within* a single Reference. While it may be tempting to establish inter-Reference linkages with this structure (e.g., when a Reference explicitly bases its concept of a taxon name on that of another Reference), for a variety of reasons I think such inter-Reference Assertion mapping is best done as a second "layer", via tables external to **tbl_Assertion**. A logical consequence of this restriction (no inter-Reference linkages from *ValidAssertionID*) is that the domain of Assertions available for entry in *ValidAssertionID* is restricted to those instances where *ValidAssertionID=AssertionID*. We can safely assume that a corresponding "valid" name Assertion will exist within the same reference that declares another name as being a "junior synonym" of that valid name (i.e., if a Reference asserts that 'B' is a junior synonym of 'A', then the same reference also asserts that 'A' represents a valid taxon).

An important concept to recognize here is that *ValidAssertionID* has context **only** for the terminal epithet name of the indicated Protonym. In other words, a Reference can treat a species-level Protonym as "valid" (*ValidAssertionID=AssertionID*), but still regard the name to belong to different genus (i.e., different combination) from the original treatment of the

Protonym. The parent-taxon context of a name within an Assertion is documented separately from its "validity", and represents the third basic element of an Assertion: the *ParentAssertionID*.

The *ParentAssertionID* links an Assertion instance to another Assertion instance that represents the parent taxon in which the first taxon is placed (according to the Reference). For example, if a Reference places species 'b' within genus 'A', then the *ParentAssertionID* for the Assertion of species 'b' will point to the Assertion of genus 'A'. As is the case for *ValidAssertionID*, I currently restrict this relationship to be within a single Reference (if a Reference explicitly places one taxon within a hierarchical parent taxon, then by implication the same Reference makes an Assertion about that parent taxon). As indicated in the diagram, *ParentAssertionID* can contain a null value. There are two reasons for this. The first is that for taxon names above the rank of species, References do not always specify what parent taxon a given taxon name is asserted to fall with (indeed, few taxonomic references explicitly state full hierarchical context all the way up to Kingdom, so at some point most references cite a taxon name without placing it within a parent taxon). The second reason is that, for Assertions made about non-valid taxon names (*ValidAssertionID≠AssertionID*), there technically is no asserted parent taxon. Although it may be assumed that the non-valid name would inherit the parent taxon of its corresponding senior synonym (the name that *ValidAssertionID* points to), I prefer instead to derive this inheritance via the *ValidAssertionID* link, and therefore I have imposed the restriction that for all cases where *ValidAssertionID≠AssertionID*, the corresponding value of *ParentAssertionID* is null. The domain of available values for *ParentAssertionID* is restricted to those Assertions of a higher TaxonRank than the current Assertion instance, and also (like the case for *ValidAssertionID*) is limited to Assertions where *ValidAssertionID=AssertionID*.

The fourth (and final) "basic element" of an Assertion is the *Epithet*. This text field stores the exact character string that the Reference used when citing the associated Taxon Name. The main purpose of this field is to document the exact spelling (including hyphens, numbers, and other symbols, where applicable) of the name within the Reference. Only the "terminal" epithet is included for binomials, trinomials, and other polynomials (including subgenera). In the special case of hybrids, the ***complete*** hybrid formula is entered, exactly as spelled and punctuated in the Reference.

One additional attribute that could conceivably be regarded as the "fifth" basic element of an Assertion is *Sequence*. The purpose of this field is to record the actual sequence in which a series of taxon names were listed, within the context of a single parent taxon. This information is sometimes useful to record, because it may represent an assertion about the phylogenetic context of a taxon among related taxa. Because the meaning of such sequence information is not standardized and its application within references is inconsistent, however, this attribute does not really constitute a "basic element" of an Assertion.

Another important attribute of an Assertion is *Pages*. In the current implementation of the model, this field is simply a text field to allow entering whatever information is necessary to designate where, within a larger Reference, an Assertion can be located. Future implementations of the model might break this information out into a separate table (e.g. linked to individual-page PDF files).

The *Reliability* index field is intended to be a semi-objective guide to how reliable an interpretation may be. Although some degree of subjectivity is inevitable in assigning this value,

I have defined the domain to be as objectively-discernable as possible, while still providing some meaningful function. A value of 5 represents the highest reliability, and is limited to only those References constituting the original description of a taxon name, or a first "new combination" Assertion. All Assertions representing Protonyms would be assigned this value. A value of 4 corresponds to other taxonomic revisionary work that explicitly treats the associated taxon within the context of the revision. A value of 3 indicates that the Reference making the Assertion did so within a taxonomic context, but not necessarily an exhaustive revisionary context. A value of 2 indicates that the Reference was scientific in nature, thought not specifically a taxonomic work (e.g., an ethological or ecological article). A value of 1 is used for popular literature and other non-scientific references. This same scale can be applied (more or less) to unpublished assertions (e.g. specimen determinations), based on the nature of circumstances and qualifications of the Agent(s) conducting the identifications. A value of 0 (default) indicates that the nature of the reliability has not been reliably determined.

The remaining three data-bearing attributes of are Assertion are provisional, and may be rendered redundant depending on how many additional Subtypes of Assertion are created. All three fields (*IsNewCombination*, *IsFirstRevision*, and *IsTypeCatalog*) are boolean values with self-evident meaning, intended to flag certain special-case Assertions which have important taxonomic or nomenclatural meaning. Any of these could be expanded to a full non-exclusive subtypes, if additional attributes relevant to each category are deemed worthy of recording.

[*CheatTaxonName* is formatted as the complete name (identical to *Epithet* for ranks of genus and higher, or complete binomial, trinomial, or other polynomial for ranks lower than genus). The value for lower-than-genus ranks is derived from recursive concatination of Epithets up to the level of genus, and the value for hybrids represents the complete hybrid formula as derived from the linkages established in the **tbl_HybridAssertion** table (see below) – which may differ somewhat from the hybrid formula as actually written in the Reference (i.e., the contents of *Epithet*, in the case of hybrids). *CheatFullTaxonName* is simply the value of *CheatTaxonName*, expanded to include all appropriately-formatted authorships. *CheatNominotypical* is simply a boolean field used to flag those Assertions that represent Nominotypical names (i.e., the *ProtonymID* values equals the *ProtonymID* value of the Assertion indicated by the *ParentAssertionID* Foreign Key). *CheatStatus* is a standardized "natural language" statement representing the combination of the four basic Assertion elements (e.g., "Valid as originally described.", "Junior Synonym of {OtherTaxonName}", "Valid {TaxonRank} within the {ParentTaxonName}", etc.]

tbl_Protonym

As mentioned earlier, **tbl_Protonym** represents a subtype of **tbl_Assertion**, indicating those special-case Assertion instances that constitute original descriptions of taxon names (i.e., Protonyms). The recursive relationship between this table and **tbl_Assertion** has been described above, and will not be repeated here. The other attributes of **tbl_Protonym**, described here, are data elements specifically associated with Protonyms (and not specifically with non-Protonym Assertions).

The Foreign Key *TypeProtonymID* is a recursive link, and is used primarily for names at the genus-group rank, to indicate which Species-group Protonym was designated as the "Type Species" for the genus-group name. I've given this field the name *TypeProtonymID* because my current thinking holds that a Type-Species to a Genus Name is a species *Name*, rather than a

species concept (generic typification is usually thought of as a nomenclatural assignment, rather than a circumscription statement). However, if a reference designates a new genus name 'A' and assigns a type species 'b' (which itself is not also described within the same reference), then there is an Assertion instance for species 'b' within this same Reference. This species 'b' Assertion instance (which, as already stated, is not a Protonym Assertion), could instead be used as the "Type Species" for the new genus 'A', in which case the type is implied to be the *circumscription* of 'b', rather than simply the *name* 'b'. In this case, the Foreign Key would be renamed as *TypeAssertionID*, and would link to **tbl_Assertion** (instead of recursively back to **tbl_Protonym**), and the rule would be imposed that the *ReferenceID* values for the corresponding instances of *ProtonymID* and *TypeAssertionID* in **tbl_Assertion** must be the same value (for a given instance of **tbl_Protonym** with a populated *TypeAssertionID*. The important question is whether a Type species of a Genus name (or Type genus of a family name) should be thought of as a *name*, or as a *circumscription*.

The Foreign Key *WordTypeID* links to the same **tbl_WordType** that was described earlier under the **tbl_Glossary** heading of the "References" section of this document. The purpose for allowing this link is to specify what word form the epithet of a Protonym takes (e.g., "Noun (apposition)", "Adjective", etc.), which can be useful for determining proper name spelling (e.g., when placed in a genus of a different gender).

The *Availability* field indicates the nomenclatural availability status of a name, as per the relevant Code of nomenclature. Values (shown in blue in the diagram) are general representations of various availability statuses, including several indicators of objective unavailability. This field represents a simple indication, which could be replaced by a more robust set of entities for tracking objective synonymy of certain names.

The *Gender* field mirrors the field of the same name in **tbl_Person** of the "Agents" section, and is used to indicate the gender of genus-group protonyms.

NomenCode indicates under which particular Code of Nomenclature a particular Protonym falls. In cases of names at ranks higher than those governed by the relevant codes, the value indicates which Code the child taxa fall under. This field is important for determing specific formatting rules of authorships, etc.

[*CheatFullProtonym* is used to store a standardized formatted name, plus authorship. The format is generally as "Epithet, OriginalParent Authorship" (e.g., "*speciesname*, *Genusname* AuthorName(s)". *CheatAcceptedAssertionID* is derived from entities and rules not shown within this section, which determine which assertion the user of the database system has decided to follow as representing the "correct" status of each name. *CheatHierarchy* is a specially-formated long text string that includes the full-context taxonomic hierarchy for each Protonym, as determind by the series of values of *CheatAcceptedAssertionID* for each name at each rank. CheatGlobalSequence is intended to be a specially-formatted text string that can be used to sort a block of taxon names into appropriate phylogenetic sequence.]

tbl_HybridAssertion

This table represents another Subtype of **tbl_Assertion** (non-exclusive with **tbl_Protonym**), that is populated with Assertions that constitute hybrid taxa. At present, the only purpose of this subtype table is to record individual Parent Taxon Assertions for a given hybrid Assertion. As with *ValidAssertionID* and *ParentAssertionID* Foreign Keys in the

tbl_Assertion table, all three values in a given instance of **tbl_HybridAssertion** (*HybridAssertionID*, *HybridParent1ID*, and *HybridParent2ID*) must point to three different Assertion records, all of which share the same *ReferenceID* source. By convention, *HybridParent1ID* stores the alphabetically-first member of a hybrid, and *HybridParent2ID* stores the alphabetically-second member. This subtype must be non-exclusive with **tbl_Protonym**, because they overlap with each other in the case of Nothospecies.

Limitations

- Because I have intentionally restricted this "Taxa" section of the data model to reflect only the "core" data elements, it does not, by itself, allow for many of data management features commonly associated with taxon concepts (e.g., mapping equivalencies of concepts or concept sets; anchoring concepts to specimens, etc.). All of these features are relatively easy to implement on top of this core data model.
- Although not a 'limitation' *per se*, it is worth recognizing thconnection between the word "Protonym" and the word "Basionym". Although "Basionym" is used primarily in botanical contexts, it could easily be extended to represent the same meaning in Zoological contexts. However, "Basionym", strictly defined, *includes* the genus-species[-subspecific] combination of names (binomial, trinomial); but only the terminal epithet is implied by the "Protonym". Moreover, whereas the word "Basionym" typically refers to the actual **name** only, "Protonym" is here extended to imply the authorship (or more directly, the Reference association) that was involved with the original establishment of the Basionym. Finally, the term "Basionym" is usually used only in the context of lower-level taxonomic ranks (genus, species, subspecies, etc.), but "Protonym" is here extended to apply to *all* taxonomic ranks. For these (and other) reasons, I have chosen to avoid using the word "Basionym" in this model, and instead substitute the word "Protonym". It may, however, later be deemed appropriate to use the word "Basionym" for this role; in part to avoid inventing new term, and in part because its primary use in this model (i.e., recovering the original full-context basionym name and associated authorship) is consistent with the more restrictive, conventional definition of the word. According to the website available at:

  http://rec-puzzles.org/new/sol.pl/language/english/spelling/nym

  "Basonym" is defined as:

  > "The earliest validly published name of a taxon, being in the case of a binomial or trinomial the source of the valid specific or subspecific epithet when the taxon is transferred to a new combination and in technical usage always accompanied by the name of the original author. (Crataegus spicata Lamark:Amelanchier spicata) [Source: Merriam-Webster's Third New International Dictionary]"

  Following this definition, the use herein would seem appropriate (although I would prefer the spelling to include the "i", to avoid additional confusion). However, to avoid confounding this issue through preconceptions about what a "Basionym" *should* mean within taxonomy, I've decided that a better approach here would be to chose a different term without all that "baggage".

  Alternative terms that might be more appropriate to use in place of "Basionym" include "Protonym" and "ProtoAssertion". The former is defined as:

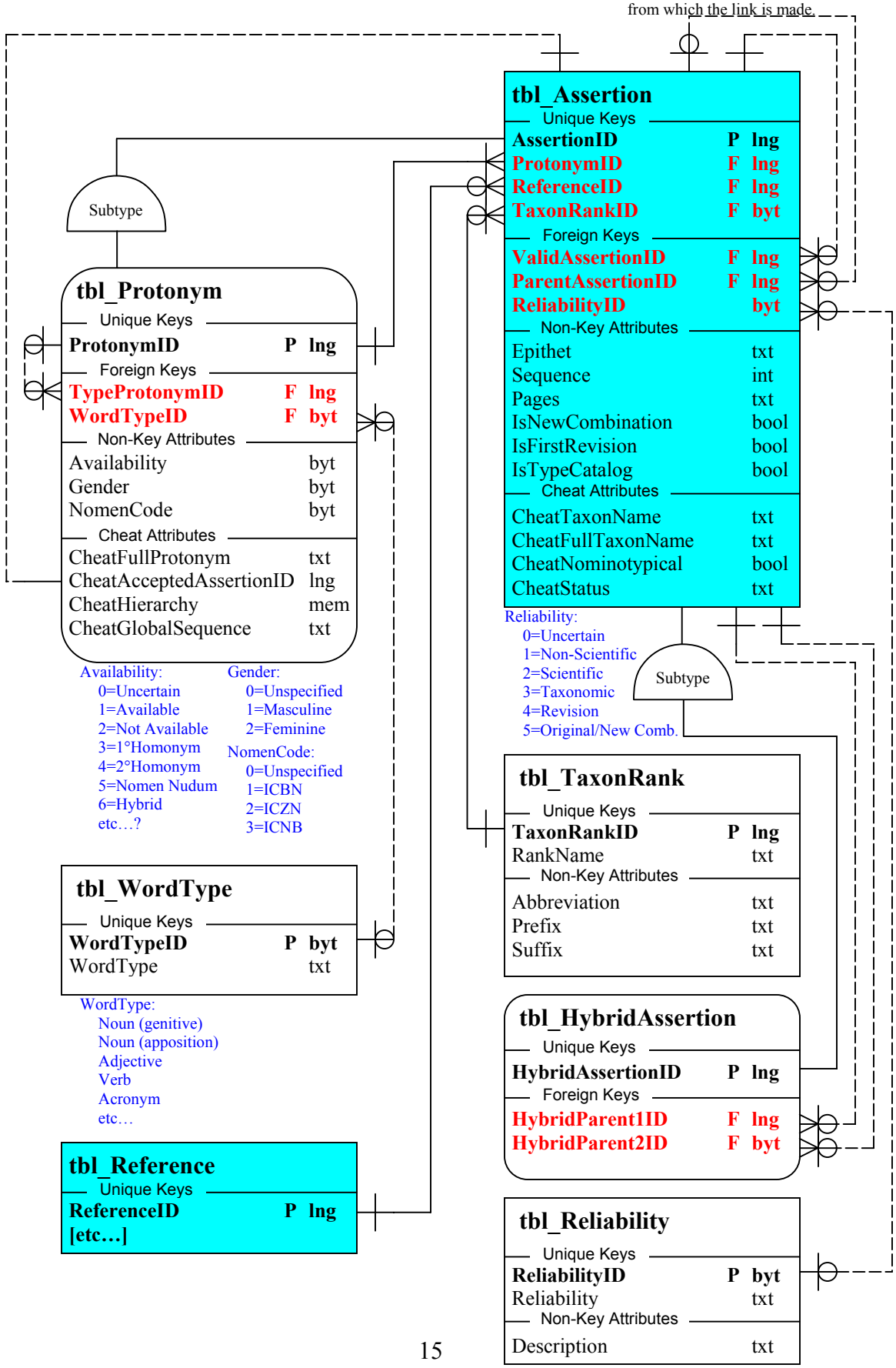
> "The first person or thing of the name; that from which another is named [Source: Oxford English Dictionary]"

Although this term is still bound by the "nym" suffix to apply strictly to a "Name" (rather than a Name-Reference intersection, as would be a subtype of **tbl_Assertion**), it seems to be a more appropriate term than "Basionym" in this context (i.e., without the baggage of preconceived meanings). It's implied meaning as a "name" per se is not entirely inappropriate, because even if it represents a subtype of an Assertion, it is intended to represent the original ***name*** component of that Assertion. An alternative solution would be to use the more explicit term "ProtoAssertion" as this would emphasize the Subtype aspect of the entity.

All things considered, I still prefer to use the word "Protonym" here. I explicitly intend for the entity it represents to be the *de facto* source of "Basionym" (*sensu* conventional usage) data.

- Another problem with the word "Basionym" is that it implies that a name has achieved legitimacy within the relevant nomenclatural Code. Strictly speaking, this would appear to restrict this model to use only with names after they have been published in accordance with relevant codes (i.e., only after they have a legitimate Basionym to point to). However, there are many applications that need to cite a Taxon Concept ***before*** it has received a Code-compliant name (Basionym). A common example of this would be specimens identified as belonging to as-yet un-named species. The use of the term "Protonym" avoids this problem.

If *ValidAssertionID≠AssertionID* for a given instance of **tbl_Assertion**, then *ValidAssertionID* must point to an instance of **tbl_Assertion** with the same value of *ReferenceID* as the instance from which the link is made.



AGENTS

Overview

The term "**Agent**" (synonymous with "Party") applies to an individual human (**Person**), or an organized group of humans (**Organization**). **AgentAssociations** may be established between any combination of a **Person**, and/or an **Organization**, and/or an **Address**. A minimum of two of these three values must be included for any single instance of **AgentAssociation** (i.e., no "association" can be made within only one of these three). For each **AgentAssociation**, there may be zero to many **EContacts** (e.g., telephone and fax numbers, telex, email addresses, websites, etc.).

tbl_Agent

Every Agent instance is assigned a *ValidAgentID* corresponding to the particular "alias" of the agent that is currently regarded as valid. If *ValidAgentID=AgentID* for a particular instance, then that specific instance represents the "most correct" variation of that Agent. If *ValidAgentID≠AgentID*, then the current Agent instance is regarded as a "junior alias" of the record indicated by the value of *ValidAgentID*. In all cases, the value in *ValidAgentID* must be drawn from the set of "valid" Agents (i.e., where *ValidAgentID=AgentID*). The *ValidAgentID* field may not contain a Null value. The *ValidAgentID* system is primarily intended to map people or organizations who have used different names over the course of their lives (e.g., maiden name and married name, organization renaming, etc.), however it is also used to record different variations of the same name for a single Agent (e.g., when a person serves as the role of Author to different publications using different sets of given-name initials, or different styles of the same multi-part last name). It is important to clarify that instances within this table do not necessarily represent a single "Agent" (Person or Organization), but actually represent various *NAMES* that have been applied to individual Agents. Unique Agents can be quickly identified as those instances where *ValidAgentID=AgentID*. This logic cascades to apply to Organization and Person subtypes.

Every instance of Agent is assigned an *AgentTypeID* value that corresponds to an existing instance of the **tbl_AgentType** table, indicating which Subtype the Agent represents. This data model currently allows only two *AgentType* values – **Person** and **Organization** – but additional *AgentType* values may be defined in the future (e.g., "Team", which would represent a set of multiple Agents who do not collectively constitute an "Organization").

An agent is flagged as *Ambiguous* if the instance does not represent a specific, identified individual Person or Organization, but rather a generic Person or Organization (e.g., "local fisherman", "fish market", etc.).

Each Agent has a *BirthDate* (or the founding date of an organization), and a *DeathDate* (or the termination date of an organization). These values are useful for distinguishing different Agents with similar or identical names.

[**CheatFullAgentName** is used store a text string representing a consistently formatted name of the Agent, for faster display in output queries.]

tbl_Organization

Organizations represent one of the defined subtypes of Agents. Conceptually, an Organization is a place-holder for the collection of individual persons who form the organization (i.e., an "organization of people"). Informal sets of multiple individual persons (e.g., a set of authors for a particular reference, or a set of collectors for a particular specimen) generally do not constitute an "Organization"; rather, organizations exist as a collection of people independently of who those particular people are at any given point in time.

Organizations can be nested hierarchically, such that any Organization might be a subset of a "Parent" organization, as indicated by *ParentOrganizationID*. Because in this implementation of the model, no form of systematic "Rank" is applied to individual organizations (e.g., "Department", "Division", "Working Group", etc.), code must be used to enforce the business rule that no organization can be its own parent, and no chain of multiple Organization→ParentOrganization links can be circular.

Organizations often have an *Acronym* and an *OrganizationName*, which are the text-strings used to represent the organization. An organization can be semi-objectively classified according to its *GeoScope*, using pre-defined values ranging from "Local" to "International" (allowing also for "Unspecified").

[**CheatFullOrganizationName** is used differently from **CheatFullAgentName**; whereas the latter provides the full name of the specific organization; the former is used to contain its full parental context. Because "parentage" is not tracked for Person-type Agents, this information is stored in this table.]

tbl_Person

The other defined subtype of Agent is Person. As explained earlier, each unique Person may be represented by multiple instances in this entity – one for each different "alias" or name variation. However, the unique individual Persons can be easily identified by filtering on cases where *PersonID* is equal to the corresponding *ValidAgentID* in **tbl_Agent** (this all applies equally to organizations).

The core fields of this table primarily involve different elements of a Person's name: *Prefix*, *GivenName*, *FamilyName*, and *Suffix*. *Prefix* and *Suffix* are straightforward, with examples given in the diagram. *GivenName* includes all elements of a person's given name, with each element separated by a space. *FamilyName* includes all elements of a person's family name (i.e., including "de", "van der", etc.). *PrimaryGivenName* is a byte-level integer representing which sequential name element of a multi-part given name is used as the primary given name. For example, for the name "John Edward Smith", the *GivenName* would be entered as "John Edward" (with a space delimiting the two given names). A *PrimaryGivenName* value of 1 would indicate that the name is formatted typically as "John E. Smith", and a *PrimaryGivenName* value of 2 would indicate "J. Edward Smith". A *PrimaryGivenName* value of 0 indicates an unspecified primary given name.

Gender indicates simply whether the person is Female (2), Male (1), or not known (0).

tbl_AgentAssociation

The primary function of this table is to track associations between Organizations and individual Persons. In most cases, this table simply serves to establish a many-to-many

relationship between people and organizations; but the function is more complex than this, because this table also serves the purpose of connecting an Association with an instance of the **tbl_Address** table. Consequently, either of the Foreign Key fields *PersonID* or *OrganizationID* (but not both) can contain a null value, but only if *AddressID* for that instance is non-null. Such an instance would allow for linking an Address directly to either an Organization or a Person, without the need to establish an Association between an Organization and a Person (e.g., a person's home address, or an organization's general address). If both *PersonID* and *OrganizationID* are non-null for a given AgentAssociation, then *AddressID* may be null for that instance (but certainly doesn't have to be).

The *AgentRole* for each instance of **tbl_AgentAssociation** is intended to represent the role played by the Person at the associated Organization. Examples are given in blue text in the diagram.

Each AgentAssociation has a *StartDate* and an *EndDate* to establish the window of time in which the association existed.

In principle, no instance should exist in the **tbl_Address** entity, unless it exists in at least one instance of AgentAssociation. Thus, the former is a "Dependent" entity of sorts, even though it serves on the "one" side of a one-to-many relationship. The fields of **tbl_Address** do not need elaboration, except perhaps for the *FmtAddress* field, which contains a fully-formatted mailing address to be entered or modified by the user. Usually, this field is automatically generated – derived from the other fields in this table – but it is not treated as a "Cheat" field because the user is allowed to over-ride the auto-formatting, to meet some particular address formatting situation. Ultimately, this is an optional, application-defined field, rather than a core field.

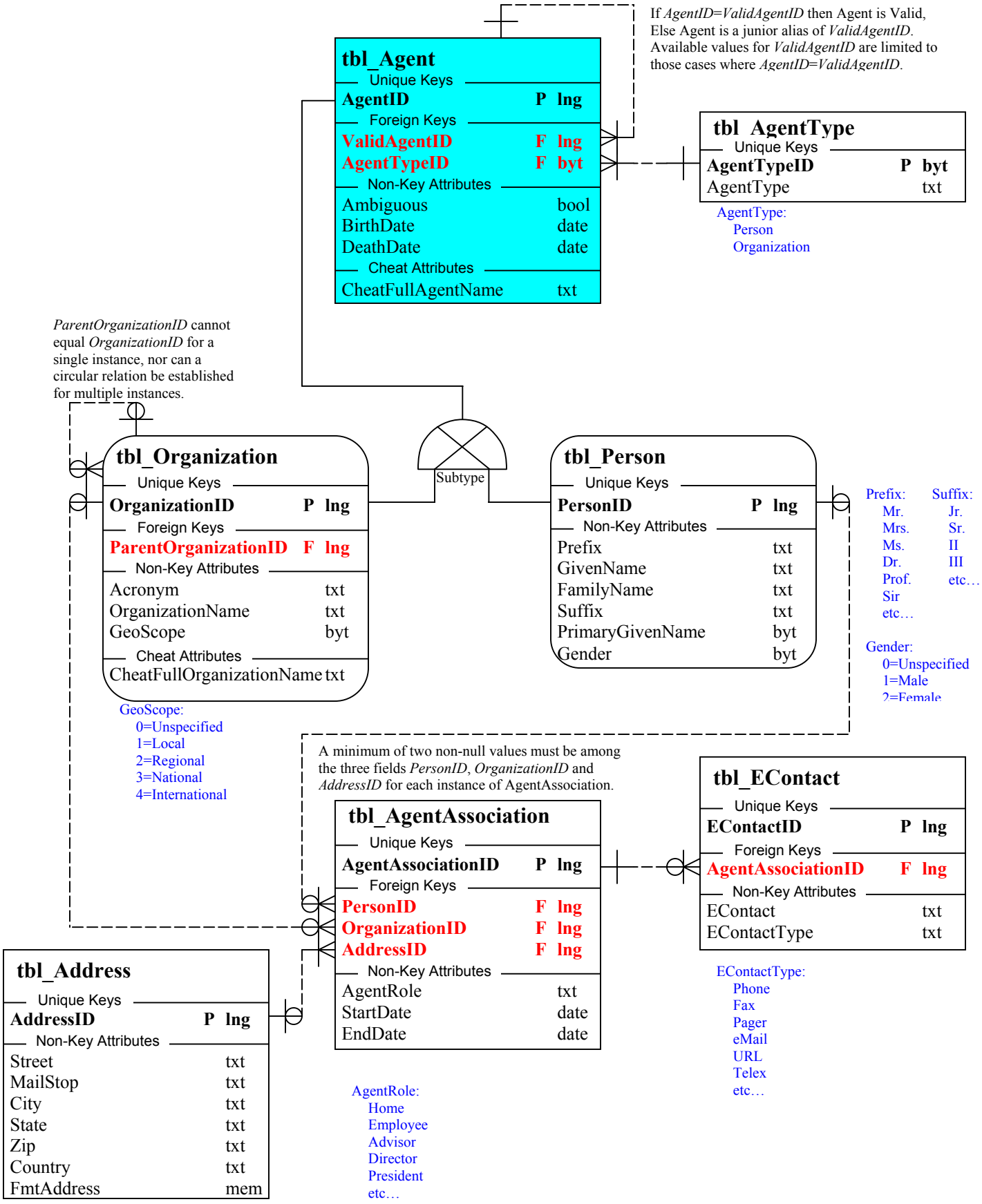
Whereas only one Address can be linked to any particular AgentAssociation, there can be many instances of the **tbl_EContact** table linked to a given AgentAssociation. The concept of EContacts represents any sort of electronic contact number or text string, such as various telephone numbers, email addresses, web URLs, etc. The type of EContact is indicated by the EContactType field, examples of which are given in blue text in the diagram.

Limitations

- Associations cannot be made directly between one Person and another Person, or between one Organization and another Organization, except for the special case of "Aliases" (by way of the ValidAgentID recursive foreign key in **tbl_Agent**), and of an Organization linking directly to a Parent Organization. Such associations (e.g., between husband and wife, or between two organizations joined by an MOU or other agreement) are considered to be outside the scope of this data model. Additional tables could easily be appended to this model to track such associations. To accommodate such relationships within the current context, one could re-define the *OrganizationID* and *PersonID* Foreign Keys of tbl_AgentAssociation to be *AgentID* and *AssociatedAgentID* (without restriction of which Subtype each is drawn from), but there would need to be structure to accommodate tracking directionality of such a relationship (perhaps in place of *AgentRole*).
- EContacts can only be linked directly to a Person or Organization (without the context of the other), if an *AddressID* has been provided for that Person or Organization. This limitation stems from the fact that **tbl_EContact** links to an instance of **tbl_AgentAssociation**, and the latter can exist only if a minimum of two of the three attributes *PersonID*, *OrganizationID*,

and *AddressID* have been populated. Relaxing this requirement of having a minimum two out of three populated foreign keys in AgentAssociation, to the more liberal rule of either *PersonID* or *OrganizationID* being populated (regardless of *AddressID*), would remove this limitation.

- Although additional AgentTypes can be defined (e.g., "Team"), they would need to be established in such a way that links to **tbl_AgentAssociation** are maintained logically. For example, if the third AgentType "Team" were established, then the *OrganizationID* foreign key of **tbl_AgentAssociation** might be redefined as "*TeamOrganizationID*" , indicating that it may be populated either with an *OrganizationID* or a *TeamID*.


If AgentID=ValidAgentID then Agent is Valid, Else Agent is a junior alias of ValidAgentID. Available values for ValidAgentID are limited to those cases where AgentID=ValidAgentID.
tbl_Agent
Unique Keys
AgentID P lng
Foreign Keys
ValidAgentID F lng
AgentTypeID F byt
Non-Key Attributes
Ambiguous bool
BirthDate date
DeathDate date
Cheat Attributes
CheatFullAgentName txt
tbl_AgentType
Unique Keys
AgentTypeID P byt
AgentType txt
AgentType:
Person
Organization
ParentOrganizationID cannot equal OrganizationID for a single instance, nor can a circular relation be established for multiple instances.
tbl_Organization
Unique Keys
OrganizationID P lng
Foreign Keys
ParentOrganizationID F lng
Non-Key Attributes
Acronym txt
OrganizationName txt
GeoScope byt
Cheat Attributes
CheatFullOrganizationName txt
Subtype
tbl_Person
Unique Keys
PersonID P lng
Non-Key Attributes
Prefix txt
GivenName txt
FamilyName txt
Suffix txt
PrimaryGivenName byt
Gender byt
Prefix:
Mr.
Mrs.
Ms.
Dr.
Prof.
Sir
etc…
Suffix:
Jr.
Sr.
II
III
etc…
Gender:
0=Unspecified
1=Male
2=Female
GeoScope:
0=Unspecified
1=Local
2=Regional
3=National
4=International
A minimum of two non-null values must be among the three fields PersonID, OrganizationID and AddressID for each instance of AgentAssociation.
tbl_AgentAssociation
Unique Keys
AgentAssociationID P lng
Foreign Keys
PersonID F lng
OrganizationID F lng
AddressID F lng
Non-Key Attributes
AgentRole txt
StartDate date
EndDate date
tbl_EContact
Unique Keys
EContactID P lng
Foreign Keys
AgentAssociationID F lng
Non-Key Attributes
EContact txt
EContactType txt
EContactType:
Phone
Fax
Pager
eMail
URL
Telex
etc…
tbl_Address
Unique Keys
AddressID P lng
Non-Key Attributes
Street txt
MailStop txt
City txt
State txt
Zip txt
Country txt
FmtAddress mem
AgentRole:
Home
Employee
Advisor
Director
President
etc…

REFERENCES

Overview

Whereas most people think of a "Reference" primarily in the context of a publication, I define the concept more broadly, in a way best described as a "Date-stamped instance of Agent(s)". All References must have as their source one or more Agents (**ReferenceAuthors**), and each instance of a Reference represents a statement by those Agents at a particular moment in time. Another way of expressing this is that a Reference may be created whenever any set of one or more Agents establishes or asserts some informational content (statement) at a certain point in time. All publications fall within this definition of "Reference", because all publications are drafted at the hand of one or more Agents (even if the Agent can only be identified as "Anonymous" or "Unspecified"), and are published at a particular point in time. Besides publications, however, there are other ways in which a set of one or more Agents may assert statements at a certain point in time. Familiar examples of unpublished References would include correspondence and other forms of personal communications (usually documented in the form of a letter, memo, or other printed documentation), and specimen determinations (usually documented in the form of specimen labels or identification tags). All other attributes of Reference deal mainly with elements of information that identify the documentation and citation details about the Reference voucher, indexing by **RefernceKeywords**, and cross-referencing References via the **ReferenceBibliography**.

tbl_Reference

The basic structure of **tbl_Reference** emulates the apparent structure of EndNote® bibliographic software. This structure was chosen to allow relatively easy transfer of Reference data between EndNote® and this database application. Several aspects of this model expand upon the basic EndNote® structure, primarily with regard to breaking certain data elements out into separate linked tables, but also in the form of extended data recording capabilities.

References may contain other references in hierarchical fashion. A familiar example would be book compiled by one set of Agents (i.e., editors), which contains chapters authored by different sets of Agents. For the purposes of this database, a more abstract and less traditional example is the ability to designate certain less-discretely-defined portions of a reference as having a different set of authors from the containing Reference. This capability is especially important for distinguishing text constituting original descriptions of taxon names from the containing reference, in cases where the authorship of the taxon name is not identical to the authorship of the containing Reference. The hierarchy of references, when it exists, is tracked by the recursive *ParentReferenceID* linkage. As with Organizations, no Reference can be its own parent, and no multiple chain of Reference→ParentReference can be circular.

Every Reference is classified according to its *ReferenceTypeID*, which is drawn from the **tbl_ReferenceType** table. Of the 19 reference types listed in blue text on the diagram, all but three ('Book Series', 'Determination' and 'Sub-Reference'), are directly mapped from EndNote. The 'Book Series' *ReferenceType* was added to accommodate citations of entire series, rather than individual volumes in a series. 'Determination' was added to accommodate the special group of unpublished References that represent taxonomic identifications of specimens. These could be lumped in with the 'Communication' (='Personal Communication') *ReferenceType*, but I decided to assign it to its own type for easier filtering. The 'Sub-Reference' *ReferenceType* is

intended to represent a portion of another, encompassing Reference (excluding cases that can be assigned to the 'Book Section' *ReferenceType*), primarily to accommodate assigning appropriate authorship to taxon names (when such authorship differs from the encompassing Reference).

Two boolean fields – *IsPublished* and *IsParent* – simply indicate which ReferenceTypes are published, and which can serve as a Parent Reference to another Reference (respectively). Additionally, **tbl_ReferenceType** contains one field for each field of **tbl_Reference** (except *ParentReferenceID*), to indicate which of the latter fields are used (and how they are used), depending on which *ReferenceTypeID* is selected for the Reference instance (see below, and also Table 1).

Depending on which *ReferenceTypeID* is selected for the particular Reference instance, there may be a link to the **tbl_ReferenceSeries** via the *ReferenceSeriesID* Foreign Key. The reference types that can be linked to a reference series include 'Generic', 'Book', 'Book Section', 'Conference Proceedings', 'Edited Book', 'Journal', 'Magazine Article', and 'Newspaper Article'. Attributes of **tbl_ReferenceSeries** are indicated in the diagram, and are not as yet rigidly defined.

Each Reference instance may be associated with the **tbl_Language** table, via the *LanguageID* Foreign Key, to indicate which language the Reference was primarily written in.

The Non-Key Attributes of **tbl_Reference** are, for the most part, drawn from EndNote's default fields. Not all attributes apply equally, or even at all, to all reference types. A matrix of how each type utilizes each field (derived directly from EndNote) is represented herein as Table 1. For simplicity, I have chosen to keep these fields in one 'flat' **tbl_Reference** table. Alternatively, they could, be broken out into different Reference subtypes; either one for each *ReferenceType*, or several clusters of one or more *ReferenceType* with similar fields (e.g., Books *vs.* Periodicals, Published *vs.* Unpublished, etc.).

Every Reference instance must be linked to one or more Agent(s) representing the Author(s) of the Reference, via the **tbl_ReferenceAuthor** entity. In cases where the specific Author is not known, a link is established to an ambiguous instance of Agent representing "Anonymous" or "Unspecified". The important point here is that a Reference is *defined* in the context of its authoring Agent(s); hence the requirement for at least one instance of **tbl_ReferenceAuthor** for each instance of **tbl_Reference**. The *Sequence* field is used to establish the sequence of authors for multi-authored references. The *ExAuthor* field is set to 'False' for all authors of all References, except those specific authors who are authors of taxon names but not authors of the Reference itself. For example, suppose a Reference is linked to ReferenceAuthors Smith, Jones, and Johnson, with the *ExAuthor* field in Johnson's record of **tbl_ReferenceAuthor** flagged 'True'. Any Protonym instance linked to this Reference (see "TAXA" section) would treat the authorship of that Protonym as "Smith and Jones (ex Johnson)". If this Reference happens to be of type 'Sub-Reference', which itself is included within a publication authored by Jones and Wilder, then the authorship for the taxon name would be interpreted as "Smith and Jones (ex Johnson) in Jones and Wilder".

As an added feature, I have included the **tbl_ReferenceBibliography** table, to record which References (*BibliographyID*) cite which other references (*ReferenceID*) in their Bibliography (or elsewhere). This can be useful in deciphering implied taxonomic concepts, to indicate whether or not one Reference had access to another Reference at the time the

Taxonomic concept was formulated. The *Sequence* field is used to establish the sequence of cited References, as they appear in the citing Reference.

[**CheatAuthors** is used to store formatted single- and dual-author last names, or first-author last name plus “et. al" for multi-authored References. **CheatFullAuthors** is used to store formatted Author names as they generally appear in bibliographies – last name and given initials for each individual author. **CheatCitation** is a concatenation of **CheatAuthors** and the *Year* field.]

tbl_Glossary

A generic system of defining words is established via the **tbl_Glossary** table. Each *Word* exists in the context of a *Language* (linked from **tbl_Language** via the *LanguageID* Foreign Key), and is assigned a *WordType* (linked from **tbl_WordType** via the *WordTypeID* Foreign Key – examples of *WordType* shown in blue text on diagram). A short *Definition* is provided for each *Word*.

Individual words can be cross-referenced to other words via the **tbl_Thesaurus** table. The nature of the relationship between the two words (e.g., ‘Synonym’, ‘Related Word’, etc.) is indicated in the *Relationship* field. Such relationships are not automatically treated as symmetrical, so in the case of a symmetrical relationship (e.g., ‘Synonym’), two instances are required in the **tbl_Thesaurus** table. Future versions of this schema may define a **tbl_RelathioshipType** table as a separate linked entity, allowing additional attributes for each relationship type (e.g., *IsSymmetrical*, etc.).

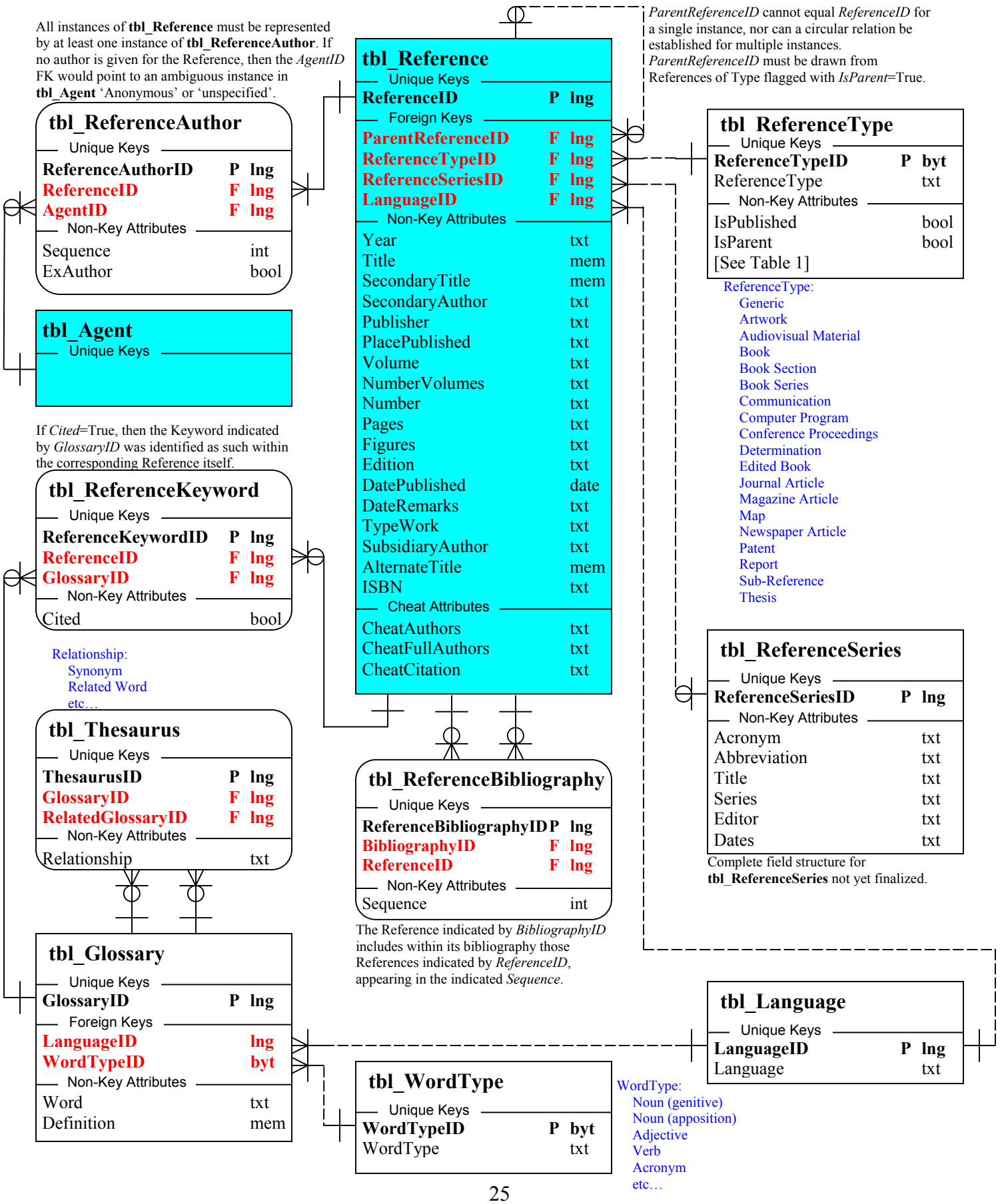
Individual instances of **tbl_Glossary** are linked to instances of **tbl_Reference** via the **tbl_ReferenceKeyword** table. If the indicated Keyword was designated in the linked Reference itself, then the *Cited* field is set to ‘True’. Otherwise, it is assumed that Keyword assignment was created by the database user.

Limitations

- The general limitation of the whole Reference structure stems from its foundation in the EndNote model. A somewhat denormalized flat **tbl_Reference** structure is taken as a compromise to maintain simplicity of import and export capability.

Table 1. Matrix of Non-Key Reference attribute uses, by Reference Type.

Generic	Year	Title	Secondary Title	Secondary Author	Publisher	Place Published	Volume	Number of Volumes	Number	Pages	Figures	Edition	Date	Type of Work	Subsidiary Author	Alternate Title	ISBN
Artwork	Year	Title	-	-	Publisher	City	-	-	-	-	-	-	Date	Type of Work	-	-	-
Audiovisual	Year	Title	-	-	Publisher	City	-	-	-	-	-	-	Date	Type of Work	-	-	-
Book	Year	Title	Parent Title	Editor	Publisher	City	Volume	Number of Volumes	Number	Pages	Figures	Edition	Date	-	Translator	-	ISBN
Book Section	Year	Title	Book Title	Editor	Publisher	City	Volume	Number of Volumes	Number	Pages	Figures	Edition	Date	-	Translator	-	ISBN
Book Series	Year	Title	-	-	Publisher	City	-	Number of Volumes	-	Pages	Figures	Edition	Date	-	Translator	-	ISBN
Communication	Year	Title	-	-	-	-	-	-	-	-	-	-	Date	Type	-	-	-
Computer Program	Year	Title	-	-	Publisher	City	Version	-	-	-	-	Platform	Date	Type of Work	-	-	-
Conference Proceedings	Year	Title	Conference Name	Editor	Publisher	Location	Volume	Number of Volumes	-	Pages	Figures	Edition	Date	-	-	-	-
Determination	Year	Title	-	-	Institution	-	-	-	-	-	-	-	Date	-	-	-	-
Edited Book	Year	Title	-	-	Publisher	City	Volume	Number of Volumes	Number	Pages	Figures	Edition	Date	-	Translator	-	ISBN
Journal Article	Year	Title	-	-	-	-	Volume	-	Number	Pages	Figures	-	Date	-	-	Alternate Journal	-
Magazine Article	Year	Title	-	-	-	-	Volume	-	Issue	Pages	Figures	-	Date	-	-	-	-
Map	Year	Title	-	-	Publisher	City	-	-	Scale	-	-	Edition	Date	Type of Work	-	-	-
Newspaper Article	Year	Title	-	-	-	City	-	-	-	Pages	Figures	-	Date	-	-	-	-
Patent	Year	Title	Published Source	-	Assignee	Country	Volume	-	Number	Pages	Figures	-	Date	-	-	-	Pat. No.
Report	Year	Title	-	Editor	Institution	City	-	-	-	Pages	Figures	-	Date	Type of Work	-	-	Rpt. No.
Sub-Reference	Year	Title	Parent Title	Parent Authors	-	-	-	-	-	Pages	Figures	-	Date	Type of Work	-	-	-
Thesis	Year	Title	Parent Title	Department	University	City	-	-	-	Pages	Figures	-	Date	Type of Work	-	-	-


All instances of **tbl_Reference** must be represented by at least one instance of **tbl_ReferenceAuthor**. If no author is given for the Reference, then the *AgentID* FK would point to an ambiguous instance in **tbl_Agent** 'Anonymous' or 'unspecified'.
ParentReferenceID cannot equal *ReferenceID* for a single instance, nor can a circular relation be established for multiple instances. *ParentReferenceID* must be drawn from References of Type flagged with *IsParent*=True.
tbl_Reference
Unique Keys
ReferenceID P lng
Foreign Keys
ParentReferenceID F lng
ReferenceTypeID F lng
ReferenceSeriesID F lng
LanguageID F lng
Non-Key Attributes
Year txt
Title mem
SecondaryTitle mem
SecondaryAuthor txt
Publisher txt
PlacePublished txt
Volume txt
NumberVolumes txt
Number txt
Pages txt
Figures txt
Edition txt
DatePublished date
DateRemarks txt
TypeWork txt
SubsidiaryAuthor txt
AlternateTitle mem
ISBN txt
Cheat Attributes
CheatAuthors txt
CheatFullAuthors txt
CheatCitation txt
tbl_ReferenceAuthor
Unique Keys
ReferenceAuthorID P lng
ReferenceID F lng
AgentID F lng
Non-Key Attributes
Sequence int
ExAuthor bool
tbl_Agent
Unique Keys
tbl_ReferenceType
Unique Keys
ReferenceTypeID P byt
ReferenceType txt
Non-Key Attributes
IsPublished bool
IsParent bool
[See Table 1]
ReferenceType:
Generic
Artwork
Audiovisual Material
Book
Book Section
Book Series
Communication
Computer Program
Conference Proceedings
Determination
Edited Book
Journal Article
Magazine Article
Map
Newspaper Article
Patent
Report
Sub-Reference
Thesis
If *Cited*=True, then the Keyword indicated by *GlossaryID* was identified as such within the corresponding Reference itself.
tbl_ReferenceKeyword
Unique Keys
ReferenceKeywordID P lng
ReferenceID F lng
GlossaryID F lng
Non-Key Attributes
Cited bool
Relationship:
Synonym
Related Word
etc…
tbl_ReferenceSeries
Unique Keys
ReferenceSeriesID P lng
Non-Key Attributes
Acronym txt
Abbreviation txt
Title txt
Series txt
Editor txt
Dates txt
Complete field structure for **tbl_ReferenceSeries** not yet finalized.
tbl_Thesaurus
Unique Keys
ThesaurusID P lng
GlossaryID F lng
RelatedGlossaryID F lng
Non-Key Attributes
Relationship txt
tbl_ReferenceBibliography
Unique Keys
ReferenceBibliographyID P lng
BibliographyID F lng
ReferenceID F lng
Non-Key Attributes
Sequence int
The Reference indicated by *BibliographyID* includes within its bibliography those References indicated by *ReferenceID*, appearing in the indicated *Sequence*.
tbl_Glossary
Unique Keys
GlossaryID P lng
Foreign Keys
LanguageID lng
WordTypeID byt
Non-Key Attributes
Word txt
Definition mem
tbl_Language
Unique Keys
LanguageID P lng
Language txt
tbl_WordType
Unique Keys
WordTypeID P byt
WordType txt
WordType:
Noun (genitive)
Noun (apposition)
Adjective
Verb
Acronym
etc…