

Sustainable operability: Keeping complex resources alive

Menzo Windhouwer^a, Alexis Dimitriadis^{a,b}

^aUniversity of Amsterdam, ^bUtrecht institute of Linguistics OTS
M.A.Windhouwer@uva.nl, alexis.dimitriadis@let.uu.nl

Abstract

The data contained in a typological database are difficult or impossible to use on their own. Sustainability must include not only preservation of the data, but also of the interface designed to present them—or a reasonable substitute. The *Typological Database System* project (TDS), which originated as a way to address issues of fragmentation and interoperability of typological databases, also points the way to a model of sustainability beyond the lifetime of a database’s host application.

1. Introduction: Obstacles to the sustainability of complex resources

While the sustainability of language resources such as corpora and dictionaries can be largely safeguarded by relying on documented, standard formats for their encoding, the approach does not scale well for resources with more complex internal structure, for which no general standard can be sufficient. Such complex resources have the characteristic that they require a certain software tool for their proper utilization; and that this software tool is not generic (e.g., an audio player, text editor, or linguistic annotation tool that supports the storage format of the resource), but is made specifically for the resource in question: Databases, in particular, are typically accessed through a custom-made user interface. A second, interacting problem is that much of the information needed to properly navigate and interpret such data is encoded in its user interface, not with the data itself. We consider the case of typological databases, and describe our approach to their integration and long-term sustainability. Consider, as a concrete example, a typological database consisting of several linked tables, accessible over the internet through a web interface comprising several forms. Numerous such databases exist today, and more are being created at a rapid pace.¹ Once they are completed, such databases are subject to the usual perils afflicting electronic linguistic resources: Gradual obsolescence of their encoding formats or host software; sudden disappearance due to incompatible software updates, retirement of a “legacy” server, or as URLs change and links fail to be updated; gradual fall into unusability with the dissipation of the insider knowledge often needed to usefully operate a poorly documented resource; etc.

To render such a database sustainable, it is not enough to export its tables in some format guaranteed to be readable (e.g., tab-separated files in a Unicode encoding, or even an SQL dump in some portable format). Doing so is insufficient in two important respects:

- a. The meaning of the table contents, and their inter-relationships, are not explicitly given in the data tables; this is the familiar problem of documentation for a resource, but exacerbated (compared to corpora or dictionaries) by the complexity and variability of database structures, and by the relatively abstract level of linguistic description involved.
- b. Even if accompanied by full documentation, a static collection of data is difficult, tedious, or even impossible to utilize without a suitable software tool. The forms and menus created by the original developers to operate a database are essential to its use, but they cannot be exported along with the data. We will term this consideration, which has not received as much explicit attention as issues of format and access, as the problem of sustainable *operability*.

To appreciate the scale of the operability problem, consider the difficulty of using a general-purpose table browser (a spreadsheet application, for example) to navigate the contents of a database consisting of several tables. Table columns (attributes) typically contain numeric values expressing different properties (whose meaning is, at best, explained in a separate document).² The tables are linked to each other by means of numeric keys with no intrinsic meaning. The process of navigating such data is tedious and error-prone, and likely to deter all but the most motivated researchers.

Lack of operability also has a detrimental impact on resource discovery: Summary metadata can only give an approximate indication of the utility of a resource for any particular task. A future researcher who will need to evaluate a large number of potentially useful resources will be hindered by the inability to inspect their contents without a large investment of effort.

1.1. The limits of data-only formats

The vast majority of existing typological databases are stored in relational database management systems. The

¹Web-accessible databases include the Graz Database on Reduplication, at <http://reduplication.uni-graz.at/>; the databases of the Surrey Morphology Group, at <http://www.smg.surrey.ac.uk/>; the Typological Database of Intensifiers and Reflexives, at <http://userpage.fu-berlin.de/~gast/tdir/>; the Stress Typology Database, at <http://stresstyp.leidenuniv.nl/>; the Berlin-Utrecht Reciprocals Survey, at <http://languageblink.let.uu.nl/burs/>; etc.

²In proper relational design, numeric values can be indices into a separate table that matches numeric codes to a text equivalent. In practice, however, the meaning of numeric values is often embedded in the user interface; and prose documentation can be non-existent or out of date.

relational structure itself is a sort of encoding standard, and would seem to provide a basis for standardization: While SQL implementations are too variable for database dumps in SQL format to be themselves portable, some version or extension of standard SQL could conceivably be chosen as the standard for data archiving. Even if the obstacles to unifying the many extant flavors of SQL could be overcome, however, the result would allow implementation-independent data storage but would still not render databases operable. The SQL schema of a database is insufficient in the same respects already mentioned:

First, it is an incomplete description of the database, since it does not include those parts of the database logic that are encoded in the user interface: Documentation and instructions to the user, business rules (explicit or implicit), and, in many cases, the text equivalents of values and menu options that are stored as small integers in the database. In the language of the OAIS Reference Model (ISO 14721, 2003), an SQL dump of a typological database is rarely “independently understandable.”³

Second, general-purpose browsers for relational databases are too low-level; they allow viewing of one table at a time, but do not automatically perform appropriate joins or aggregations of records in one view—and, even with knowledge of foreign key declarations, have no way of determining which joins or aggregations are “appropriate.” Simply put, the user interface of a database is underdetermined by its relational schema.

We doubt that these problems are restricted to relational databases. Similar issues doubtless arise with other complex resources developed with their own interface, and with other data models besides relational databases.

1.2. Toward a solution

The difficulty of achieving sustainable operability can be summarized as follows: Complex resources require ad hoc software that cannot be maintained over the long term; so operability can only be ensured by relying on generic software that can be maintained, and periodically replaced, in a cost-effective manner. But traditional data archiving practices do not provide enough information for generic software (or even human specialists in many cases) to reconstruct the proper structuring and presentation of the data.

It can be seen now that to fully meet the goal of sustainable operability, the archived data must first be “independently understandable.” We can distinguish here between user-oriented metadata (documentation), which helps users interpret the data when it is presented, and formal, system-oriented metadata that is machine-understandable and can describe not only the encoding and relational structure (narrowly considered), but also appropriate ways of managing and presenting the data to the user.

³The OAIS Reference Model charges conforming archives with ensuring that archived information be “independently understandable” by its designated target community, i.e., interpretable without recourse to hard-to-access resources, including the individuals who created it. This is considered necessary for long-term data preservation. We thank an anonymous reviewer for calling this point to our attention.

What is needed, minimally, is a software platform that provides operability of typological databases with diverse structures. While no tool could probably be fully generic and at the same time achieve operability without a prohibitive amount of configuration, the problem is not intractable when restricted to one application domain at a time—in our case, to the data models applicable to typological databases. But no software platform can make up for the lack of information that is essential to managing or understanding a resource; this problem must be addressed by ensuring that the required information is collected, and is suitably utilized by the software platform in question.

Sustainable operability, in short, requires two things: sufficiently rich metadata and documentation for the data to be not only “independently understandable” by its end-users, but also for automatically determining appropriate ways of rendering it; and a software tool, or a series of software tools over a long period of time, that utilize this information to provide the actual operability.

To provide operability of an open-ended collection of resources in a practical way, there must be a way for a generic application (or several) to be used with all of them. Because the native storage formats (usually relational) are insufficient to describe typological databases to a degree that allows operability via a generic tool, we adopt a hierarchical, semi-structured data model that combines the data itself with rich documentation of database contents and of the linguistic properties being described. We will term this the *Integrated Data and Documentation Format (IDDF)*. Sustained operability is then a matter of mapping resources to the IDDF format at the time of archiving, and maintaining a generic tool, or tools, that support searching and browsing over IDDF resources. This approach accomplishes operability of the databases in the narrow sense, and also provides access to the documentation needed by the end user to properly interpret the available data.

Eventually, even the generic software will approach obsolescence due to changes in web technology, host operating systems, and the like. At that point it will need to be replaced by new IDDF-aware software with analogous functionality. The self-describing nature of IDDF documents is meant to support their migration to new access tools (or even the addition of new tools next to existing ones) without any changes to the resources themselves.

But long-term operability is more than a matter of keeping the software running. A proper solution should also support other considerations of sustainability. In particular, it should be positioned within a scenario involving data archiving and its complement, resource discovery.

The Typological Database System (TDS), described in more detail in section 2., is a working implementation of such an architecture. The TDS provides integrated access to a collection of independently developed typological databases through a single, generic web interface. Databases are imported into the system through a process that combines rich documentation of all aspects of the data with automated transformation of the data itself into a common, hierarchical data space. The result is a unified data structure (the IDDF data tree) that can be searched or

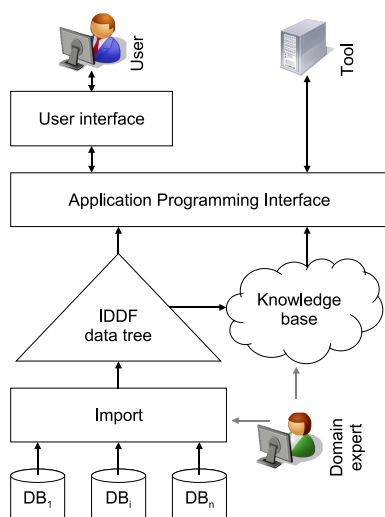


Figure 1: The TDS architecture

browsed over the web through the TDS webserver.⁴

While the process can easily be performed on each database separately, the approach has the added benefit of allowing the integration of multiple databases into a unified resource. (This is in fact the primary goal of the TDS). Arguably, integration is not essential for sustained operability of the resources; but it greatly enhances their usefulness, efficiency of utilization, and ease of resource discovery.

Data archival inadvertently exacerbates the problem of operability, because archives cannot commit to long-term hosting and maintaining a kaleidoscope of diverse database applications; rather than wait for obsolescence of the software or hardware, operability threatens to be lost at the moment of uploading the static content of a resource to a digital archive. From our perspective, this can be seen as a blessing in disguise: Sustainability problems can be addressed while the original technical infrastructure is still operational, and the custodians of a resource still possess the required knowledge (either in their heads or as offline documentation).

2. The Typological Database System

The Typological Database System is a web-based service that provides integrated search access to a collection of independently developed typological databases. The system consists of a data integration module and a web server that provides access to the integrated data.⁵ At the intersection of the two parts is the IDDF, a hierarchical data model that integrates data and metadata from multiple databases into a unified data space.

Figure 1 shows the TDS architecture. The primary data input to the system comes from the component databases.

⁴<http://language.link.let.uu.nl/tds/>.

⁵The TDS is a project of the Netherlands Graduate School of Linguistics (LOT). It is supported by a grant from the Netherlands Organization for Scientific Research (NWO), and by funds from the participating universities (University of Amsterdam, Utrecht University and Leiden University). For more information on the TDS, see (Saulwick et al., 2005; Dimitriadis et al., 2005; Dimitriadis et al., 2008).

A domain expert creates an import schema that includes a mapping of each database into a unified hierarchy, enriched by documentation of the data and its relationship to the common TDS knowledge base. On the basis of this schema, data and documentation from multiple databases are integrated into a single hierarchical structure, the *IDDF data tree*. A separate component of the system, the TDS webserver, supports querying, browsing, and resource-discovery functions over the collected data.

The entire system is XML-based and relies on a number of (commercial) open source or freely available libraries. It is written largely in Java, XSLT, XQuery and a XML pipelining language specific to the application server 1060 NetKernel.⁶

With around a dozen databases currently in the TDS, the total number of parameters in the system is well over a thousand; hence the system follows a two-stage access model, consisting of resource discovery followed by query formulation and execution. During the resource discovery stage, users search or browse the combined metadata to discover database fields of interest. The user interface supports integrated search, display and navigation of the metadata, presenting users with the information necessary to assess both the relevance and the correct interpretation of a field. Selected fields are accumulated using a shopping basket model. In the second stage, the user constructs and executes a query on the basis of the fields in the query basket.

2.1. The integration process

The import schema is defined in a special-purpose language developed by the TDS project, the *Data Transformation Language* (DTL).⁷ The TDS import engine interprets the DTL specifications, and uses an appropriate software plug-in to extract data from a copy of the original database (which can be in a variety of database formats) and transform it into the IDDF tree.

Typically, the documentation provided with a database is insufficient to make its semantics and logical structure fully explicit, and the creation of the DTL specification involves repeated interaction between the TDS domain expert and the creators of the database. The required metadata, which often lives only in the heads of the database's creators, is in this way elicited and recorded. The process is non-trivial but necessary for the sustainability of the data. Because the developers of the component databases have devoted much time and effort to collecting information in their databases, each component database represents a valuable resource; and therefore the time investment is justified.

In any event the process is reusable: Once the transformation schema has been defined, new data added to the database can be imported with minimal human intervention. In this way a database can be mapped to the IDDF before the data collection is finished and its data frozen.

⁶<http://www.1060.org/>.

⁷The DTL is a non-procedural language that allows an IDDF schema to be specified and annotated, and the resulting data tree to be populated from the database contents. It was designed for use by linguists with no special technical background. See (Saulwick et al., 2005; Dimitriadis et al., 2008).

Only if the database schema is modified is it necessary to modify the transformation schema.

It should be added here that while it is necessary to have a working understanding of a database’s semantics in order to integrate it into the TDS, much of the documentation collected and recorded into the IDDF tree is not explicitly encoding-related, but intended for the benefit of the end-user. For example, a TDS component database gives the number of basic color terms in some languages as “4.5”. As a matter of encoding it is enough to know, as its documentation explains, that color term counts can be fractional numbers, and that 4.5 means “between four and five”. But what does “between four and five” *mean*? It might indicate a dialectal split, inconsistencies between speakers, the presence of a marginal or dubious color term, uncertainty about the facts, or all of the above. The answer is of interest to potential users of the database, and only its creators can provide it.

Conceptually, the DTL is just one means of carrying out this transformation;⁸ what is important from our present perspective is that the DTL, or an equivalent, defines a mapping of a data resource into an IDDF tree; and that the result comprises a combination of data and relevant documentation. Our vision of the IDDF is as an open format, which can be generated and manipulated by other tools. Section 3. gives more details on its structure, and on the way other components of the TDS architecture can be generalized.

2.2. What is transformed

Independently created data resources differ in a variety of ways, which need to be addressed during the integration process. The TDS makes an important distinction between differences in encoding (in the broad sense) and differences stemming from deeper theoretical or practical considerations. The former include variation in font encodings or notational conventions such as interlinear gloss labels, codes for Boolean values (*true/false* vs. *0/1*, etc), the organization of information into fields and tables, etc. The deeper differences are ultimately differences in meaning (semantics): They stem from considerations such as the theoretical commitments of a research group (including the associated terminology), the specific classificatory categories and coding decisions adopted during the construction of a database, etc.

While standardization efforts might one day lead to more uniformity in structure and encoding among databases, they will have no effect on the divergence of theoretical viewpoints and research traditions that constitutes the most intractable source of heterogeneity. These diverse viewpoints are not only dearly held by their practitioners: They are the subject matter and outcome of linguistic analysis, and cannot (should not) be replaced by any uniform, agreed-upon framework. While it might seem like a good idea to transform data into some “standard” terminology, the abstract nature of typological data collections makes this impossible. First of all, two theoretical terms are rarely if ever exactly co-extensive; even if they were, the terminology

⁸One could, for example, convert data into XML and transform it by means of hand-written XSLT, as the TDS did during the pilot phase of the project.

```
<iddf:warehouse
  xmlns:iddf="http://.../ns/iddf">
  <iddf:meta>
    <iddf:scope id="tds" type="warehouse">
      ...
    </iddf:scope>
    <iddf:notion id="n1" name="language"
      scope="tds" type="root"
      key-datatype="enum">
      <iddf:label>Language</iddf:label>
      <iddf:description>
        One of the world's languages
      </iddf:description>
      ...
    </iddf:notion>
    ...
  </iddf:meta>
  <iddf:data xmlns:tds="..." ...>
    <tds:language iddf:notation="n1"
      key="...">
      ...
    </tds:language>
    ...
  </iddf:data>
</iddf:warehouse>
```

Figure 2: The top-level structure of the IDDF.

adopted by a researcher is often the result of a deliberate process, and can be felt to be as much a part of a linguistic analysis as its empirical claims. To substitute terminology under such circumstances would be a form of misrepresentation.

Accordingly, the TDS approach is to compensate for encoding differences wherever possible, by transforming the source data to adhere to, or at least be relatable to, a uniform design (“object model”); but semantic divergences are maintained, and are made explicit by suitable documentation and careful construction of relationships between various levels of metadata.

Because the various component databases each have their own schema and focus, i.e., they are heterogeneous, the aggregated IDDF data is semi-structured. To assist in the process of resource discovery by end-users, the TDS metadata includes links to a unified knowledge base, consisting of an ontology of linguistic terms and several taxonomies that provide quick domain-oriented entry points.

3. Sustainable operability with the IDDF

At the heart of the TDS, and of our vision for sustainable database operability, is the IDDF data tree. It organizes data and metadata into a unified structure that provides sufficient information for generic resource discovery, query operations, and interactive browsing tools.

3.1. The IDDF data structure

The IDDF data structure consists of two parts, a metadata schema and a data part. The metadata part defines and annotates the schema to which the data part conforms.⁹ We use the term “data tree” to refer to the entire structure, since the

⁹The IDDF can be conceptually considered as the concatenation of two documents. The document as a whole is valid XML,

two parts are closely interrelated. An abbreviated example is shown in figure 2. (A detailed example is given in the Appendix).

Figure 3 provides an informal overview of the conceptual structure of the IDDF data tree. It can be informally understood as a hierarchy of nodes (called Notions), which serve a variety of functions.

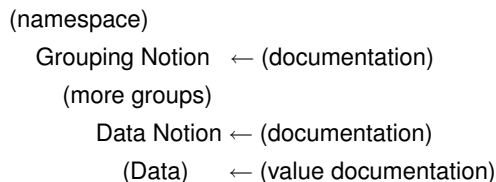


Figure 3: Conceptual organization of the IDDF data model.

At the leaves of the tree are *Field Notions*, which correspond to fields of the component databases.¹⁰ When the tree is built (“instantiated”) by importing the databases, these Notions are populated with the data. (Note that the documentation remains in the schema portion of the IDDF, as shown above).

There are also *Grouping Notions*, which contain other Notions (either of the data or the grouping kind) and thus define the hierarchical structure of the IDDF data tree. Fields from several databases can be mapped to the same part of the tree, even the same Notion; for example, the attribute *Language Name* is a single Notion used for all databases. (The TDS organizes data according to topic, regardless of its database of origin; one could easily adopt a different policy, and map each database into a dedicated part of the hierarchy).

To facilitate management of all this data from diverse sources, Notion definitions are overlaid with a system of namespaces, which can be nested; Notions defined in a particular namespace can only be used within its scope. For example, the TDS project defines a top-level “tds scope” that provides the upper levels of semantic context, such as *clause-level phenomena*. The component databases can then define database-scoped Notions as descendants of appropriate points in the global hierarchy.

Besides its content, each Notion is associated with documentation and format information (which are stored in the schema part of the IDDF, as detailed below). Grouping Notions can be associated with a description of the kind of data they dominate, including summaries of the linguistic theory and terminology of the data providers; Field Notions can be associated with a description as well as an enumeration of possible values, which can themselves have associated documentation.

validated against a Relax NG schema that essentially ignores the data section. Validation as an IDDF document requires two passes: After the initial minimal validation, an XSLT 1.0 stylesheet is run on the metadata section to generate a complete Relax NG schema. This is then used to validate the entire IDDF document.

A sample IDDF document, and the required schema and stylesheet, are available at <http://language.link.let.uu.nl/tds/iddf/>.

¹⁰The relationship to the original database fields is not one-to-one. Some Notions are in fact created by splitting up or combining several database fields.

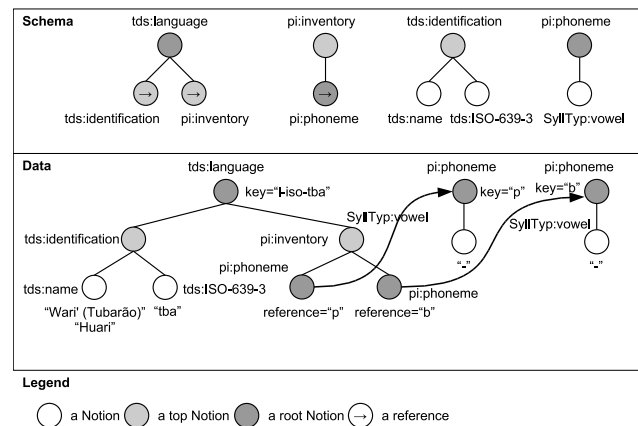


Figure 4: Graphical representation of the example IDDF schema and data tree

In many cases, a database uses a number of fields for information that belongs together and should be considered as a whole. For example, geographic latitude and longitude together make up geographic coordinates, and these together with language name, ISO code, and other essential information make up the *Language Identification* group. Each such group of fields is mapped to a subtree of the IDDF, which is identified as a *semantic context* by means of a special label assigned to its root Notion. These *Top Notions*, as they are called, are treated specially by the TDS search and browsing interface.

Larger hierarchies can be built by reusing these semantic contexts and nesting Notions inside each other. There can be multiple separate hierarchies, each with its own top-level root (called a *Root Notion*). Hierarchies can be linked to each other by establishing a primary/foreign key relationship between a Root Notion and another Root Notion. The role of Root Notions in the IDDF data model can be compared with tables in the relation model.

Figure 4 shows how the hierarchical definitions in the schema, `tds:language`, `tds:identification`, `pi:inventory` and `pi:phoneme`, are utilized during the instantiation process of the data tree. The reference leaves indicate the valid ways of linking these hierarchies together; e.g., `pi:inventory` is nested in `tds:language`; through the `pi:phoneme` reference in `pi:inventory`, the hierarchies `tds:language` and `pi:phoneme` are related.

Each of these building blocks, i.e., Notions, scopes and values, can be extensively described in the metadata. The metadata part of the IDDF document shown in the Appendix starts with describing four scopes: `tds`, `pi`, `SyllTyp` and `UPSID`. Due to space limitations, we do not discuss scopes further. A Notion schema can contain the following information:

1. an identifier;
2. a scope;
3. (optional) a label;
4. (optional) a description, possibly formatted using XHTML;
5. (optional) one or more typed links to the knowledge base;
6. (optional) one or more links to other Notions;

7. (optional) semantic data type;
8. (optional) semantic value data type and/or key data type
9. (optional) an enumeration, possibly partial, of the possible values or key values; and for each (key) value:
 - (a) the literal (key) value as it appears in the data;
 - (b) (optional) a label;
 - (c) (optional) a description;
 - (d) (optional) one or more links to the knowledge base;
 - (e) (optional) one or more links to other Notions.

The example in the Appendix includes several Notions that illustrate some of these documentation units: `tds:ISO-639-3` has a description marked up with XHTML to include a link to the standards website; `pi:phoneme` and `SyllTyp:vowel` have links to concepts in the ontology, such as *segment* and *vowel*; `pi:inventory` has the semantic data type UPPC (Universal Phoneme Positioning Chart, see (Dimitriadis et al., 2008)); the metadata of Root Notions `tds:language` and `pi:phoneme` contain enumerations of their possible key values, while `SyllTyp:vowel` contains an enumeration of its values (see Figure 5).

```
<iddf:notion id="n7" name="vowel"
  scope="SyllTyp">
  <iddf:label>Vowel</iddf:label>
  <iddf:description>
    Is the segment a vowel?
  </iddf:description>
  <iddf:link type="concept" rel="as"
    href="http://...owl#vowel"/>
  <iddf:link type="concept" rel="to"
    href="http://...owl#vocalicFeatureNode"/>
  <iddf:values datatype="enum">
    <iddf:value>
      <iddf:literal>+</iddf:literal>
      <iddf:description>
        The segment is a vowel.
      </iddf:description>
    </iddf:value>
    <iddf:value>
      <iddf:literal>-</iddf:literal>
      <iddf:description>
        The segment is not a vowel.
      </iddf:description>
    </iddf:value>
  </iddf:values>
</iddf:notion>
```

Figure 5: Example of IDDF metadata associated to a notion.

3.2. The data

Since there are multiple top-level Root Notions, the data tree is actually a forest of trees, each of them an instantiation of a hierarchy dominated by a Root Notion. These trees are linked to each other using the *key* and *ref* attributes (see the Appendix). As Notions (with the exception of Top and Root Notions) can't be uniquely identified by just the combination of the scope and the identifier, each node in the tree also specifies which Notion is being instantiated, using the `iddf:notion` attribute.

Each instantiation is based on data from at least one component database. The source of a node in a tree is indicated by the `iddf:srcs` attribute. When data loaded from various databases are in agreement, they are instantiated as a single node and this attribute lists all these database scopes. But databases may also disagree. For example the Syllable Typology Database uses the name “Wari” (Tubarão) for a certain language, while UPSID uses “Huari.” Both names are stored in the IDDF document, but each in its own `iddf:value` node with a `srcs` attribute indicating its origin.¹¹

3.3. The IDDF surroundings

3.3.1. The metadata and data source

The IDDF, as already mentioned, is an ordinary XML format. There are no barriers to creating valid IDDF documents with tools other than the DTL engine; one might wish, for example, to design a description language with a different syntax and primitives, perhaps for resource types that are very different than the typological databases we have been working with. Another possibility might be for a (complex) database application to directly support IDDF as an export format, without the intervention of a description language. In this case, the descriptive metadata might still need to be manually supplemented. This indicates that there could be a need for specific IDDF metadata editors. It is easy to visualize the use of a specific GUI to annotate Notions, and perhaps even to create the semantic hierarchies (contexts).

3.3.2. Links to external semantic resources

As figure 1 shows, the IDDF document can be linked to a knowledge base. In the case of the TDS this consists of an OWL ontology, developed during the course of the project, and a number of SKOS taxonomies. This allows the TDS to semantically extend queries by following the formal relationships in the ontology. The taxonomies provide alternative organizations of entry points into the data schema. Other forms of encoding knowledge, e.g. in the form of a tag cloud, could also be associated with the IDDF schema. In the TDS project, developing the metadata and the knowledge base went hand in hand. In applications of IDDF where the metadata is readily available one could also extract the knowledge base, or part of it, by mining the metadata (Feldman and Sanger, 2006; Cimiano, 2006). To get enough input for the mining algorithms one might use other related inputs, e.g., in the case of scientific databases the articles written on the basis of the data. One could also bootstrap the mining process by manually creating an initial domain-specific knowledge base.

3.3.3. Standards

Because the data in typological databases is overwhelmingly about languages, data aggregation depends crucially on reliably identifying the language that data is about. The TDS protocol relies on ISO 639-3 language codes (ISO

¹¹Note that the IDDF could have also allowed each database to be mapped to a separate hierarchy, avoiding any chance of an overlap or clash.

639-3, 2007), internally and externally, to identify the language described and carry out data integration. ISO language codes are used internally as part of the key, and they are always utilized for data integration, if available. For databases or records that do not provide them, the TDS domain experts attempt to add them (by means of the DTL script), on the basis of language names and the assistance of the database creators. Again, this is a labor-intensive process but is justified in view of the value of the data, and unavoidable if the language described is to be unambiguously identified. (Once again the result is enrichment of the original data through the transformation process). In alternative application domains where cross-database integration of records is not a goal, such issues are less of a concern.

To control the proper handling of the various kinds of integrated data, the IDDF tracks the data type of each variable; the primitive types *free text* and *enumeration* can be overlaid with an open set of other (semantic) types, which are defined dynamically in the IDDF schema (that is, through the DTL) and typically apply to a group of related Notions rather than to a single one. The TDS web interface, for example, has special renderer modules for the semantic types *interlinear glossed text* (consisting of aligned morphemic tiers, a translation, etc.) and *phoneme inventory*¹²

To fully exploit this approach, it should be possible for Notions (atomic or complex) to be associated with standard data types or controlled vocabularies. Thus the ISO language code can be linked to the namespace of the appropriate authority, which provides a controlled vocabulary shared by other tools; fields conforming to other controlled vocabularies can be linked to the appropriate “data category” registered in the future ISO Data Category Registry (ISO 12620, 2008; Kemps-Snijders et al., 2008), etc. Other encoding types such as MIME types, complex structures like interlinear glossed text, etc., should similarly be reported in a standard way, and/or linked to an appropriate URI to allow their identification.

In effect, this approach extends the notion of standard data types beyond simple numeric, text and enumerated types, to more complex aggregations of data. There still work to be done in the domain of registering such resource types (the ISO Data Category Registry is designed to cover only unitary data types, not hierarchies), but the IDDF can be positioned to utilize such advances when they occur.

4. The generic user interface

The rich structure of the IDDF has made it possible to develop a generic data browser service for the typological database domain, available through the TDS server.

The TDS server is divided (somewhat imperfectly, at the moment) into an Application Programming Interface (API) and a web interface. While the web interface is closely tied to the state of today’s web browsers and associated technology (including JavaScript support, etc.), the API is considerably more stable. By untangling these two better, an API can be created that provides services to multiple generations of other tools.

¹²The phoneme inventory type triggers a specific table-based rendering of a full or partial phoneme inventory.

The data browser is generic, in the sense that it does not incorporate schema or data information about any of the component databases; all such information resides in the IDDF. The browser is limited, however, by the kind of data models and displayable objects one expects to find in typological databases. Much of the data in typological databases can be displayed as tables of short values, and therefore such tables are prominent in the browser interface. There are special provisions for presenting interlinear glossed text and tables of phonemic inventories, and a mapping module for displaying data values at the geographic location of the language in question. On the other hand, there is currently no provision for displaying video streams, or (more importantly) any provision for managing data aligned to particular portions of a video stream.

While more such display modules can be developed as necessary, the browser remains generic only in the limited context of the intended application domain. For very different kinds of resources (such as experimental measurements, corpora, annotated multimedia data, etc.), one can imagine a completely different data browser that is suited to the structure of that application domain. The IDDF itself can encapsulate a wide variety of such formats.

The structure of the IDDF also makes partial compliance possible: An IDDF-aware tool, for example, could extract and manipulate interlinear glossed text from a larger resource whose full structure is not supported by the tool.

Finally, it must be acknowledged that the TDS interface (and probably any conceivable generic equivalent) is not as effective in presenting data as the best custom-built typological database interfaces; but it is more than sufficient for providing operability of the data, and other generic browsers over the IDDF data could undoubtedly do even better. In any event, several of the component databases of the TDS had no autonomous interface at all, or only a very primitive one; and the TDS interface is immensely more effective than these.

5. The IDDF in broader context

The issues we have discussed are not new, of course. We have already mentioned OAIS, the Open Archival Information System Reference Model (ISO 14721, 2003), which provides definitions of terms related to data archiving and defines roles and responsibilities in the context of a functional model. The OAIS document discusses in some detail the requirement that archived resources should be *independently understandable* to their target community of users, and also acknowledges the issue of operability, mentioning that the native user interface sometimes encodes information essential for its understandability and noting that “maintaining Content Information-specific software over the Long Term has not yet been proven cost effective due to the narrow application of such software.” In this context, our approach can be seen as a way to achieve an economy of scale, by transferring the burden of operability to domain-wide generic tools which manage the generic IDDF format. This will reduce the burden of maintaining operability in a very scalable way, and will *hopefully* prove to be acceptably cost-effective. Whether this expectation will be realized can only be determined in the long term.

The OAIS also devotes attention to issues of archiving for the *Long Term*, defined as a period long enough to raise issues of adapting to new technology or a changing user community. The latter issue, of a changing user community, is not one we address directly; our user-oriented documentation is intended to make data independently understandable to present-day linguists, not future ones. However, there is sufficient creativity and variation in today's linguistic theories that even for understandability by contemporary linguists, they must be documented in some detail. Thus the documentation that is necessary today will serve as a good basis for understandability in the future.

Mapping a database to IDDF format requires manual enrichment of the resource with metadata that cannot be automatically computed from its schema. Typically, the creators or maintainers of the original resource are asked to provide supplementary information (concerning both formal and user-oriented metadata) that is not embedded in the native data dump. While this is necessary if the resource is to be independently understandable (and is therefore indispensable to real data preservation), it means that the approach is applicable only to data of sufficient value to merit this sort of intervention. For very large-scale data collection projects, this kind of attention to each incoming resource might well be impossible. In such cases, the IDDF architecture can still support operability at a lower level, comparable with that provided by present-day solutions: The resource, along with whatever documentation is available, is imported in a form that simply mirrors the relational structure of the original database. Such data cannot be rendered in the most appropriate way, but can be browsed and manipulated at the relational table level by suitable generic software. This gives a level of functionality equivalent to viewing a database with a DBMS administration tool.

For large-scale data integration, then, the IDDF “dumbs down” to a level of functionality comparable to that provided by some existing large-scale archiving solutions. For example, (Heuscher et al., 2004) addresses the task of archiving the records of the Swiss Federal Administration, which are reported to be growing at a rate of some twenty terabytes per year. The SIARD project achieves “software-invariant” archiving of relational databases via transformation, at time of import, to a consensus SQL model (SQL-3). “On principle, functionality (i.e. software, hardware) is not archived” (Heuscher et al., 2004, p. 1). Archived data can be browsed at the relational table level by reloading into a conforming DBMS. The Chronos system (Brandl and Keller-Marxer, 2007) maintains data in its original dump format and provides low-level user access, again at the level of browsing the relational structure and tables, by supporting “on-the-fly migration” from an ever-growing collection of dump formats. This approach, while allowing archives to be maintained on a very large scale, does not provide high-level operability, especially for complex data of the type we have been concerned with. The IDDF architecture allows higher levels of operability to be achieved where this is practical, but can be (under)utilized to yield low-level operability for large volumes of complex data.

Roles and responsibilities

The architecture described relies on software support at two levels: On the input side, there must be tools to support the creation of IDDF documents. On the access side, there must be a generic data browser for any supported application domain. The two levels of tools have different maintenance requirements:

Once a resource has been mapped to the IDDF format, input-side software is not needed for its continued operability (unless, of course, the original resource changes and needs to be re-imported). An archive that stores resources in IDDF format need only ensure the continuous availability of appropriate data browsers on the access side. As such browsers become outdated or unmaintainable, they must be replaced by new IDDF-aware browsers with analogous functionality.

For IDDF-based archiving to be practical, however, suitable conversion tools are necessary. In the TDS architecture, IDDF generation is carried out by the TDS import engine, which is driven by DTL schemas and relies on plugins that grant it access to various database and dump formats.¹³

In principle, responsibility for maintaining IDDF generation tools (or using them) need not rest with the archive. A resource provider can arrange to export their data in IDDF format, perhaps via a DTL-like transformation module or in some other way. If the format should become widespread, one could even expect general-purpose DBMS applications to support such conversions. For the meantime, however, archives relying on the IDDF architecture must also address the problem of bringing data to IDDF form.

6. Conclusions

As we have seen, the problem of sustained operability of complex resources is ultimately traceable to the limitations of common storage and interchange formats, which do not provide sufficient information for generic navigation. By focusing on the particular (but broad) domain of typological databases, we have shown that the rich IDDF architecture can integrate sufficient information for a generic data browser adapted to the types of data common in typological databases. The approach is extensible and suitable for alternative application domains, as long as there is some homogeneity in the kind of data that is being collected (regardless of how each resource has chosen to present it). In effect, the idea of storing resources in a standard format that can be managed with generic tools is extended to families of complex formats that represent similar data collections. A notable aspect of the TDS is its focus not only on metadata pertaining to encoding formats and operability, but also on documentation intended for the end-user. Because of the abstract nature of linguistic analysis, such user-oriented

¹³Note that while a diverse collection of such formats must be specifically supported, there is no need to support long-obsolete formats. When an archive no longer plans to archive databases stored on eighty-column punched cards, there will be no need to maintain support for this format (or the associated hardware). Once a resource is converted to IDDF, the original format is irrelevant to operability.

documentation is essential for the proper interpretation of high-level resources like typological databases.

More generally, by collecting and centralizing metadata and documentation, the TDS archival procedure safeguards the interpretability (and therefore true operability) of the archived data.

In the context of an archival environment, the IDDF architecture also solves the problem of versioning and citeability of evolving resources: Instead requiring resource creators to maintain multiple versions of their database, an archive can simply host multiple versions of a resource, and make them available (and operable) as if they were separate databases. Hence the archive can provide a versioned, operable mirror of the database without the need for any versioning provisions in the database schema itself.

In short, the rich IDDF format can support sustainable operability of complex resources, by allowing a critical mass of such resources to be managed through generic (but domain-specific) tools.

7. References

- Stefan Brandl and Peter Keller-Marxer. 2007. Long-term archiving of relational databases with Chronos. In *First International Workshop on Database preservation (PresDB '07)*.
- Philipp Cimiano. 2006. *Ontology Learning and Population from Text*. Springer-Verlag, Berlin.
- A. Dimitriadis, A. Saulwick, and M. Windhouwer. 2005. Semantic relations in ontology mediated linguistic data integration. In *Proceedings of the E-MELD Workshop on Morphosyntactic Annotation and Terminology: Linguistic Ontologies and Data Categories for Linguistic Resources*, Cambridge, Massachusetts, July.
- A. Dimitriadis, M. Windhouwer, A. Saulwick, R. Goedemans, and T. Bíró. 2008. How to integrate databases without starting a typology war: The Typological Database System. In S. Musgrave and M. Everaert, editors, *The Use of Databases in Cross-Linguistic Studies*. Mouton de Gruyter. To appear.
- Ronen Feldman and James Sanger. 2006. *The Text Mining Handbook*. Cambridge University Press.
- Stephan Heuscher, Stephan Järman, Peter Keller-Marxer, and Frank Möhle. 2004. Providing authentic long-term archival access to complex relational data. In *European Space Agency Symposium "Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data"*.
- ISO 12620. 2008. Terminology and other language resources – Data categories – Specification of data categories and management of a data category registry for language resources. To appear.
- ISO 14721. 2003. Space data and information transfer systems – Open archival information system – Reference model.
- ISO 639-3. 2007. Codes for the representation of names of languages – Part 3: Alpha-3 code for comprehensive coverage of languages.
- M. Kemps-Snijders, M. Windhouwer, P. Wittenburg, and S.E. Wright. 2008. ISOcat: Corraling data categories in the wild. In *Proceedings of the International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May.
- A. Saulwick, M. Windhouwer, A. Dimitriadis, and R. Goedemans. 2005. Distributed tasking in ontology mediated integration of typological databases for linguistic research. In *Proceedings of the International Workshop on Data Integration and the Semantic Web*, Porto, Portugal, June.

Appendix: A longer IDDF example

We include here a sample IDDF structure. The first part (<meta/>) integrates data schema and documentation, while the <data/> element contains the sparse data.

```
<iddf:warehouse
  xmlns:iddf="http://.../ns/iddf">
  <iddf:meta>
    <iddf:datatype id="UPPC"/>
    <iddf:scope id="tds" type="warehouse">
      <iddf:label>
        Typological Database System
      </iddf:label>
      <iddf:scope id="pi">
        <iddf:label>
          Phoneme Inventories
        </iddf:label>
        <iddf:scope id="SyllTyp" type="database">
          <iddf:label>
            Syllable Typology Database
          </iddf:label>
        </iddf:scope>
        <iddf:scope id="UPSID" type="database">
          <iddf:label>
            UCLA Phonological Segment
            Inventory Database
          </iddf:label>
        </iddf:scope>
      </iddf:scope>
    </iddf:scope>
    <iddf:notion id="n1" name="language"
      scope="tds" type="root">
      <iddf:label>Language</iddf:label>
      <iddf:description>
        One of the world's languages
      </iddf:description>
      <iddf:keys datatype="enum">
        <iddf:key>
          <iddf:literal>
            l-iso-tba
          </iddf:literal>
          <iddf:label>Aikan&#227;</iddf:label>
        </iddf:key>
        ...
      </iddf:keys>
      <iddf:notion ref="n2"/>
      <iddf:notion ref="n5"/>
    </iddf:notion>
    <iddf:notion id="n2" name="identification"
      scope="tds" type="top">
      <iddf:label>
        Language identification
      </iddf:label>
      <iddf:notion id="n3" name="name"
```

```

    scope="tds">
<iddf:label>Name</iddf:label>
<iddf:values datatype="free"/>
</iddf:notation>
<iddf:notation id="n4" name="ISO-639-3"
  scope="tds" >
  <iddf:label>ISO 639-3 code</iddf:label>
  <iddf:description
    xmlns:xhtml="http://...">
    The code as assigned to the
    language in the
    <xhtml:a href="http://...">
      ISO 639-3 standard
    </xhtml:a>.
  </iddf:description>
  <iddf:values datatype="enum">
    <iddf:value>
      <iddf:literal>tba</iddf:literal>
    </iddf:value>
    ...
  </iddf:values>
</iddf:notation>
</iddf:notation>
<iddf:notation id="n5" name="inventory"
  scope="pi" type="top"
  datatype="UPPC">
  <iddf:label>
    Phoneme inventory
  </iddf:label>
  <iddf:notation ref="n6"/>
</iddf:notation>
<iddf:notation id="n6" name="phoneme"
  scope="pi" type="root">
  <iddf:label>Phoneme</iddf:label>
  <iddf:link type="phoneme" rel="as"
    href="http://...owl#segment"/>
  <iddf:keys datatype="enum">
    <iddf:key>
      <iddf:literal>p</iddf:literal>
    </iddf:key>
    <iddf:key>
      <iddf:literal>b</iddf:literal>
    </iddf:key>
    ...
  </iddf:keys>
  <iddf:notation id="n7" name="vowel"
    scope="SyllTyp">
    <iddf:label>Vowel</iddf:label>
    <iddf:description>
      Is the segment a vowel?
    </iddf:description>
    <iddf:link type="concept" rel="as"
      href="http://...owl#vowel"/>
    <iddf:link type="concept" rel="to"
      href="http://...owl#vocalicFeatureNode"/>
    <iddf:values datatype="enum">
      <iddf:value>
        <iddf:literal>+</iddf:literal>
        <iddf:description>
          The segment is a vowel.
        </iddf:description>
      </iddf:value>
      <iddf:value>
        <iddf:literal>-</iddf:literal>
        <iddf:description>

```

```

    The segment is not a vowel.
  </iddf:description>
</iddf:value>
</iddf:values>
</iddf:notation>
</iddf:notation>
</iddf:meta>
<iddf:data
  xmlns:tds="http://.../ns/iddf/tds"
  xmlns:pi="http://.../ns/iddf/pi"
  xmlns:SyllTyp="http://.../ns/iddf/SyllTyp"
>
  <tds:language iddf:notation="n1"
    key="l-iso-tba"
    iddf:srcs="SyllTyp UPSID">
    <tds:identification iddf:notation="n2"
      iddf:srcs="SyllTyp UPSID">
      <tds:name iddf:notation="n3"
        iddf:srcs="SyllTyp UPSID">
        <iddf:value srcs="SyllTyp">
          Wari' (Tubar&#227;o)
        </iddf:value>
        <iddf:value srcs="UPSID">
          Huari
        </iddf:value>
      </tds:name>
      <tds:ISO-639-3 iddf:notation="n4"
        iddf:srcs="SyllTyp UPSID">
        <iddf:value
          srcs="SyllTyp UPSID">
          tba
        </iddf:value>
      </tds:ISO-639-3>
    </tds:identification>
    <pi:inventory iddf:notation="n5"
      iddf:srcs="SyllTyp UPSID">
      <pi:phoneme iddf:notation="n6" ref="p"
        iddf:srcs="SyllTyp UPSID"/>
      <pi:phoneme iddf:notation="n6" ref="b"
        iddf:srcs="UPSID"/>
      ...
    </pi:inventory>
  </tds:language>
  ...
  <pi:phoneme iddf:notation="n6" key="p"
    iddf:srcs="SyllTyp UPSID">
    <SyllTyp:vowel iddf:notation="n7"
      iddf:srcs="SyllTyp">
      <iddf:value srcs="SyllTyp">
        -
      </iddf:value>
    </SyllTyp:vowel>
  </pi:phoneme>
  <pi:phoneme iddf:notation="n6" key="b"
    iddf:srcs="SyllTyp UPSID">
    <SyllTyp:vowel iddf:notation="n7"
      iddf:srcs="SyllTyp">
      <iddf:value srcs="SyllTyp">
        -
      </iddf:value>
    </SyllTyp:vowel>
  </pi:phoneme>
  ...
</iddf:data>
</iddf:warehouse>

```