Design Considerations for DDI-Based Data Systems

by Alerk Amin[1], Ingo Barkow[2], Stefan Kramer[3], David Schiller[4], and Jeremy Williams[5]

Abstract

Growing amounts of available data and new developments in data handling result in the need for advanced solutions. Therefore, organizations providing data have to focus more and more on technical and design issues. In order to keep the effort and expense low, data storage and data documentation must go hand in hand. This paper aims to help decision-makers by highlighting two promising approaches - relational databases for data storage and the DDI (Data Documentation Initiative) standard for data documentation. Possible interactions between both solutions are discussed, whereby the focus is on the advantages and disadvantages of representing DDI in its native XML format vs. the storage format of relational databases. In addition, three use cases are presented to provide further clarity on design considerations for DDI-based data systems: (1) agencies with existing relational database structures, (2) agencies with homogeneous DDI input and output, and (3) agencies with mixed environments..

Keywords

Data Documentation, Data Storage, Compatibility, DDI, Relational Data Base, SQL

Introduction

Data constitute a valuable, perhaps the most valuable, commodity in scientific research. Therefore, the potential for reusing generated data for future projects is an important consideration in the conduct of research (see Pienta 2010). But data can only be reused if they can be sufficiently interpreted and understood, and that requires that they be well documented (see Gregory et al. 2009).

Data reuse is not the only reason for proper data documentation in the social and behavioral sciences. When contextual information for a given dataset is captured effectively, new opportunities for comparative research are enabled across multiple datasets. International comparative research requires harmonized data and a cross-national standardization of data documentation. In Europe, the project "Data without Boundaries" (DwB), funded by the 7th Framework Program of the European Commission, is developing a framework to facilitate research within the member states of the European Union (http://www.dwbproject.org). Furthermore, there is a growing demand for merged datasets from different data sources resulting from the improvement of statistical methods and technical capabilities (see Lane 2010). All these developments result in the fact that multiple data providers are involved in the process of data generation for scientific research. Nevertheless, even if only data from one survey is needed, normally

> The main advantages of a rules-based model is the certainty and lack of ambiguity

more than one organization is involved due to the fact that data collection, data preparation and data dissemination are often undertaken by different partners (or at least different departments). These new developments mean that multiple data providers are often involved in the process of data generation for scientific research. And increasingly, even for single surveys, more than one organization is involved due to the fact that data collection, data preparation, and data dissemination are often done by different partners (or at least by different departments), as shown in Figure 1.

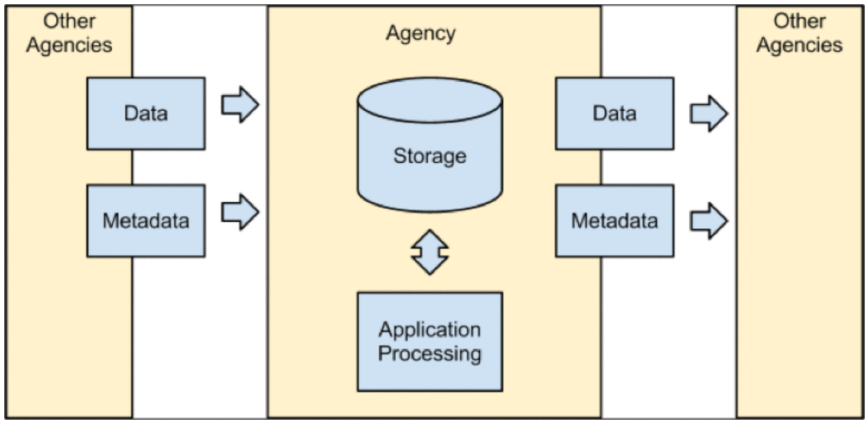

Figure 1: Multiple agencies play a role across the data pipeline.

To address all of these challenges, and to enable sound scientific research in the future, a documentation standard for research data is vital. The DDI[6] metadata specification (see Vardigan et al. 2008) offers a solution; many important data providers are already using DDI, or are about to use it. The DDI Alliance[7], which develops the DDI specification and promotes its worldwide adoption and implementation, is supported by an active community that steadily works on improvements.

The DDI standard provides a means to represent metadata about data collected in the social sciences, and potentially other disciplines (see Block et al. 2011), in a meaningful and structured manner. It is therefore expressed using XML as a framework. XML stands for eXtensible Markup Language[8]. It offers rules for a human- and machine-readable format for data exchange. XML schemas are employed to structure metadata content in the form of DDI instances.

Essentially, DDI can represent metadata in the form of XML files based on the DDI XML Schema[9]. The XML files can be stored on a common file share, or can be put into an XML database like (BaseX[10] or eXist[11]) to enable collaborative work with multiple users. Another possibility is to represent DDI in relational databases (RDBs). The table below describes the basic organizational difference between an XML hierarchy and a relational model. While XML unfurls its content, like a tree structure, from one top level down to the most detailed content, a relational store models information in tables with non-hierarchical contents. These tables are bound together via defined keys (in the example below, see "organization_scheme_id").

It is obvious that a standard like DDI can only serve the scientific community, and provide solutions for the mentioned challenges, if it is actively used by a sufficient number of stakeholders. In order to achieve this goal, the DDI-based documentation must be easy to understand and easy to integrate into existing data infrastructure. It also has to be compatible with future developments in the area of data storage. Relational databases are a widely used and flexible solution for data storage. Bringing DDI together with the capability of relational database systems will promote both data storage for the purpose of scientific research and adoption of DDI as a standard.

This article is based on two related papers authored by Amin et al. in 2011 and 2012 and written with the software developer community in mind. To complement these more technical papers on the usage of DDI in relational databases, this paper is oriented towards management in agencies using DDI or considering its use. The paper discusses the advantages and disadvantages of representing DDI in relational databases as an alternative to an XML structure. In addition, several short use cases are provided to inform the task of decision-making for DDI-based system design.

Relational Databases versus XML: Pros and Cons

The idea of storing DDI instances in a relational database, as opposed to an XML database, is often a hot topic among developers. From the perspective of DDI solely as a "storage" standard, an XML database has certain advantages. But when thinking of DDI as a transport format between applications, the actual storage format for each application should be the one that best meets that application's needs. In many cases, a relational database is the better option. The following section of the paper demonstrates the advantages of using a relational database.

Representing the DDI model within a relational database

The first reason to consider a relational database model for DDI arises from an organizational point of view. Many agencies have been storing primary data and associated metadata for time spans measured in decades, and a very common storage method is the relational database, as its tabular structure is ideal for storing rectangular data resulting from data collection activities. Therefore, those agencies have a high level of expertise and investment in using the relational database model. Changing their present table-based metadata standard (whatever that may be) to a DDI representation which is also

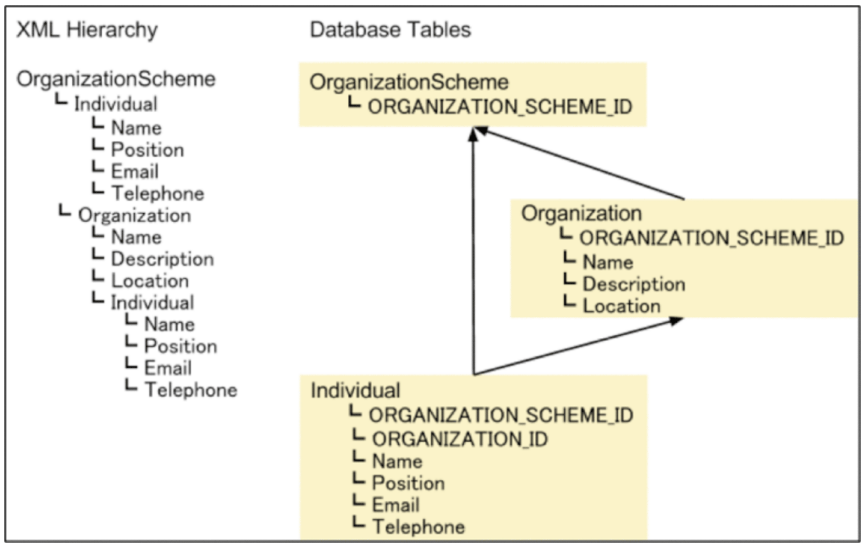

Table 1: Comparison of XML and RDB structures.

table-based should thus be intuitive to them. Using XML for storage, on the other hand, might be problematic, as these agencies do not have the experience or resources to convert the metadata and change the surrounding tools to the new structure. XML may be known to them, but mostly as an import or export format. They might therefore be reluctant to utilize DDI in XML format for reasons of transformation costs or because they may have to leave their area of expertise or comfort.

In addition to organizational considerations, there are also structural advantages to using a relational database. Agencies often represent their microdata internally in the form of a relational database as a central storing mechanism because it is ideal for processing rectangular data (e.g., SPSS data files, ASCII data files) in tables and can manage the file structures of multiple studies by input and output processes. If the metadata are stored in the same database as the microdata, the movement from metadata to data output works seamlessly, as native database methods such as connecting tables by referential integrity can be used. The metadata can be linked to the associated research data. A user can therefore first search the metadata and then move easily to the connected data. This model can even be extended to create custom data extracts (like a variable shopping basket), where an extract of the dataset, including the related metadata as a kind of codebook, can be selected and downloaded, e.g., via a Web interface. In an XML-based DDI environment this can also be done, but with much more effort, as two different structural models have to be merged. In a worst case scenario, an external service has to link between an XML metadata structure based on DDI and an ASCII file containing the microdata.

Relational databases have existed on the market for decades, and have led to the development of many tools for working with them. If one extends the idea of combining metadata and microdata into a relational database model, then the next step can be changing the database model into an analytical one. Relational databases can be enhanced to become analytical or multidimensional databases (e.g., online analytical processing [OLAP] cubes[12]). With this model, enhanced analytical or statistical methods from the area of Business Intelligence (e.g., data mining, process mining) can be applied to the data. These methods might lead to completely new research questions and new knowledge. This change of model would be difficult to realize in a purely XML-based environment.

A less complex example is storing more than one survey in a structure. In a relational database, the tabular structure can be designed to support multiple surveys in one structure by adding additional administrative tables. In a DDI structure based on XML files, this is difficult to represent; and it is difficult in an XML database, as the structure is largely based on the original DDI XML schema, which normally (as it is file-based) demands a separate XML file for each survey. In an XML database structure each survey on its own has to be represented as a separate XML database or at least as a separate instance of an XML database (if the XML database supports instances). The problem can be solved by adding additional programming routines surrounding the XML structure to emulate referential integrity by XML database linkage. Nevertheless, the relational database offers these possibilities intrinsically or with much less effort.

A representation of metadata within a relational database can also be independent of the DDI version and/or instance. Some agencies use an internal structure for their metadata that is not based on DDI but contains all the necessary information needed to exchange data with other agencies. For them, DDI in its XML form can be used as an import and export format. For example, ICPSR offers an "Export Study-level metadata" (in DDI 2.1 or 3.1, as of Oct. 2011) function for studies in its data archive[13] in this manner. A possible advantage of this method would be that the surrounding processes can always be adapted to the desired or required DDI version(s), which is far less challenging than updating native DDI XML instances to the appropriate version. Nevertheless, a major drawback of relational databases importing XML file structures is the possibility for information loss. If some nodes within the XML instance have no representation within the database structure, this content will simply be lost during the import process, or the import will not work at all if there is a structural check disallowing these kinds of partial imports.

Representing the DDI model in XML instances

Although the relational database contains a lot of additional features, the "native" way to represent DDI content is to store DDI as an instance specified by the XML schema. This leads to the logical advantage of a direct representation of the content in the correct schema. A DDI instance using the full set of DDI elements will be far superior to a construct within a relational database, as not all functionalities of DDI can be represented easily in the latter. Problems arise with a relational database, as will be shown further below, in representing versioning in DDI (see Edwards et al. 2009), or pointing to another agency by using referential URNs. In native XML the solution can be quite easily expressed, but in relational databases this is possible only with heavy additional programming (e.g., incrementing versioning by surrounding Web services or using analytical databases with slowly changing dimensions to represent the time or version). However, most agencies do not use DDI in its full specification, but only a small subset of elements; here, the advantages of the XML approach may not weigh heavily. Essentially, if an agency uses the full DDI specification, the XML database storage implementation is superior as this is the best possibility to express DDI as designed by the DDI Alliance.

Issues with DDI specification changes in relational databases and in XML

A problem all implementations of DDI share is handling new versions of the specification (e.g., DDI 3.1 to DDI 3.2). If a new version of DDI is extended with new structures, or there are changes in the structure itself, this causes significant problems in implementation. In the case of the DDI-RDB, this means constructing a new import and export mechanism for the new version. Furthermore, it might lead to a change in the overall database model to support both versions. In a worst case scenario, the structures are not compatible anymore, leaving the organization with two different databases or at least database partitions for storing the information, which is a considerable problem in data management.

But the DDI-XML method faces challenges with specification changes as well. Either the DDI-XML structure has to be transformed, or there have to be multiple versions of DDI-XML in the XML database. This leads also to changes in the application logic of the associated tool. If one chooses the simple solution from above and only changes the nodes which are known to the agency, again this leads to the inconsistency problems mentioned before.

A sizable advantage for the DDI-XML representation here is its hierarchical structure. DDI-XML is capable of expressing complex structures in an organized manner and can use built-in XML features like inheritance or validation against the schema. A DDI-RDB has to use additional program logic to emulate this behavior. In some cases, inheritance can only be expressed by using complex join operations

between tables or self-joins within a table, leading to a decrease in speed while accessing the information. These performance issues can be ameliorated by using advanced database optimization techniques like partitioned view, partitioned tables, or managed code, but in the end there is still a structural disadvantage.

Considering a hybrid RDB-XML database approach for DDI

Another way to keep the imported XML structure intact without losing performance or without logical losses would be to use the XML features of commercial databases. Some database systems, such as Microsoft SQL Server 2008 R2[14] , Oracle 11g[15], and IBM's DB2,[16] have added support for managing XML natively within the cell structure of their tables. This includes advanced features like XML indexes, XML data type (thus XML will not be handled as string, but recognized as XML), and XPath search expressions within table cells. Furthermore, it is also possible to link from an XML database like eXist to a relational database with similar results.

Using the hybrid approach, the advantages of relational databases (e.g., multiple studies, high performance) can be combined with the flexibility of XML databases and enable easier handling of DDI between different systems. Currently, several agencies are experimenting with this approach.

Use Cases for Handling DDI in Different Database Structures

While the aforementioned papers (see Amin et al. 2011) focused on the usage of DDI in relational databases and were targeted more at the software developer community, this paper highlights the management considerations needed to choose the right infrastructure for different agencies. The following use cases provide illustrations to that end.

Use Case 1 - Agency with existing relational database structure

In many cases, agencies focusing on DDI are not completely new in this domain, but have engaged in data management and curation for a long time and already have existing studies. Therefore, because of the possibilities of handling large amounts of data in a structured form, relational databases are quite common in those agencies, and a lot of knowledge and familiarity with relational databases already exists. Nevertheless, this might also be the case with XML databases depending on the agency. Therefore, the decision of which model to use (relational, XML, or hybrid) largely depends on the established structure of a given agency. If an agency uses only relational databases, the preferred choice in most cases would be to stay in the same environment. This means the people responsible for the database design have to check which elements that already exist in their table structure can be matched to DDI to create import and export mechanisms. If, for example, an agency already has a table to store items for their instruments, the agency needs to determine whether all the necessary fields (or at least the mandatory ones from the DDI element QuestionItem) are present. If not, it can check whether these fields exist in other tables of the database model or can be extracted from additional material which can then be loaded into the database by import mechanisms.

If the agency also obtains datasets from third parties, this ingest process becomes more complicated. Based on the elements the other agency used, the database scheme has to be checked to see if the table structure can be enhanced by using elements from DDI which do not yet exist in the database before an import can happen. These checks must occur, at a minimum, with the elements that are mandated by the DDI schema. In the next step, the tools using the relational database have to be adapted to use the additional tables within the enhanced database structure. It is, in most cases, not necessary for an agency to implement all the elements of DDI into the relational database structure. DDI consists of a multitude of elements which are not necessarily applicable for all agencies.

In some instances, an existing application may have a structure that is too far removed from the DDI model. For these applications, incorporating the DDI model into the existing database may not be a suitable option.

For these situations, the following may be better options:

- Creation of a completely new relational database using established elements and DDI elements. The former metadata can be imported by Extract-Transform-Load (ETL) processes. This involves an automated work flow to export the data from the old database, process it in a staging area, and finally move it into the new structure.
- Creation of a hybrid database using the old database for established elements and combining them with additional DDI information from another database (this can also be XML instead of a relational model). This option can make the programming of the surrounding tools quite complex, as it utilizes multiple paradigms and languages in tandem with each other.
- Starting from scratch only for new studies. With this option, the agency decides to use a new database model (can be relational or XML for DDI representation) and also develops new tools for it. Older studies are still handled in the old model.

Use Case 2 – Agency with homogeneous DDI inputs and outputs

The appropriate use of native XML databases is beyond the scope of this paper. It is worth mentioning, however, that there are many cases that would warrant using a native XML database rather than a relational database. One such use case is that of a system that ingests a homogeneous instance of DDI, perhaps generated from a software tool, and outputs the same, possibly for the purpose of implementing search functionality or for long term preservation of DDI in its original format. For the purpose of this example, homogeneity is defined as using the same version of DDI. Some organizations are able to achieve uniformity by mandating a standard for metadata throughout the organization by instituting common tools, practices, etc. If the same version of DDI is being used throughout an organization (represented as XML), then a native XML database offers reduced system complexity resulting in lower conversion costs and higher output fidelity.

One necessary component of any relational database-driven DDI application is mapping a given instance of DDI to a relational structure. Implementing this step provides all of the benefits mentioned elsewhere in this paper, but it does add complexity to the architecture of a system. If DDI XML is also the output format, which is likely, then the mapping would have to occur again, this time from the relational structure to DDI XML.

By using a native XML database, the DDI XML ingested can simply be stored and queried in its original format, bypassing the mapping step altogether and reducing system complexity. Reduced complexity means lower conversion costs, and often means a lower probability for errors occurring, which will naturally

increase output fidelity, or the probability that the XML input into the system is the same as that which is provided as output.

These benefits can also be accomplished using a relational/XML hybrid approach, and would likely be preferred due to the increased flexibility of blending the relational and hierarchical paradigms.

Use Case 3 – Agency with a mixed environment

In most situations, an application that uses DDI will not exist in isolation. Using DDI promotes reuse and long-term preservation. This will usually require coordination between multiple applications, often residing at different agencies.

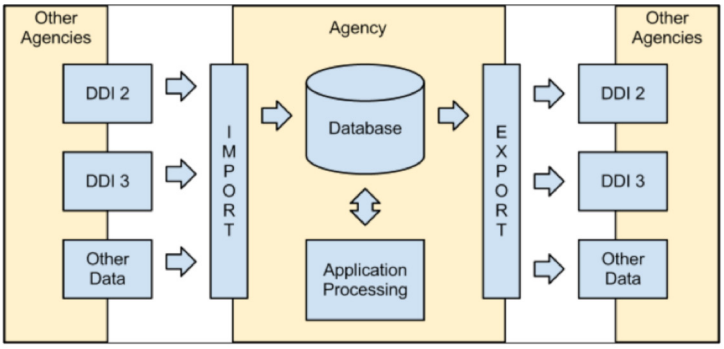

Figure 2: Importing and exporting metadata in DDI format

In a heterogeneous environment, the inputs and outputs for a particular program are vitally important. As a standard, DDI can help with this. Two applications or agencies can use DDI to communicate metadata. For a limited scope, two applications can use a single version of DDI. But for a larger system, being able to support multiple versions of DDI becomes important.

When designing an application, there are three factors to consider when thinking about interoperability: the inputs to the application, the outputs from the application, and the logic of the application itself (what the application does with the data internally). In most cases, the application logic is the "value added" by the application/agency, and is often the most important part of the system. The needs of the application logic often dictate the other requirements of the system.

When dealing with multiple sources of data, there can be multiple formats or standards. Even for agencies that use DDI, there can be different versions (DDI Lifecycle - "DDI 3" in the above graphic - and/or different versions of DDI Codebook - "DDI 2" in the above graphic). One option is to write an application layer that deals with all of these different formats, but this is usually a poor option. It involves multiple copies of the application logic to deal with each format, and a new version of the application logic must be written for each future version of DDI. And while this may help with reducing the import complexity, it does not solve any problems during export.

A better option (see Figure 2) is to use import modules to transform the various incoming formats into a single format for internal storage and processing. This allows for a much simpler application logic, because it can be written to process a single data structure that is appropriate for the logic. Additionally, new formats can be dealt with by writing new imports, without having to change the application logic. For exporting, transformation modules can be used to convert the internal data to the appropriate format.

Looking forward: DDI version 4

Whereas the DDI metadata specifications through version 2.1 have been expressed as Document Type Definitions (DTDs), and starting with version 2.1 through version 3.2 as XML schema, the forthcoming version 4 will be represented in OWL/RDF as well as XML schema. Work on implementing the DDI metadata model using Semantic Web standards had begun at a workshop in September 2011 (http://www.dagstuhl.de/11372), with an early focus on how best to relate Resource Description Framework (RDF)-described datasets to other related resources and objects (publications, geographies, organizations, people, etc.) in the Semantic Web (see Kramer et al., 2012). In October 2012, the DDI Alliance established the Moving Forward project to create a model-based specification for DDI. The ongoing work, and the structure of the DDI 4 model, are documented in a wiki (see DDI Alliance, 2014). This development will enable new methods to derive appropriate database schemas for a particular subset of all items. In DDI ver. 4, there will be "views" - a subset of the complete model (e.g., for a particular codebook format). These views could form the basis for the creation of a database schema.

Conclusion

The data deluge that the scientific community is experiencing combined with new developments in data handling result in the need for advanced solutions. Therefore, organizations disseminating data have to focus more and more on technical and design issues. In order to keep the effort and expense low, data storage and data documentation have to go hand in hand. A more flexible usage of DDI tailored to relational databases could ease some of the challenges. Whereas storing DDI in XML database provides certain advantages for data documentation systems, the flexibility of a relational database storage model can more appropriately answer some of these challenges.

Decision-makers have to keep in mind that they need to enable future-proof solutions which result in uncomplicated use of data for scientific research. This paper has only given some first hints. More research has to be done in the area of data documentation and data storage. One promising topic of conversation could possibly be found in a hybrid RDB-XML database approach for DDI. A lively discussion on this and other topics would be timely and well received. Therefore, the authors invite discussion of this paper on the DDI users' email discussion list[17], and at future meetings of the DDI and broader social science data communities.

References

Amin, A., Barkow, I., Kramer, S., Schiller, D., Williams, J. (2011). Representing and Utilizing DDI in Relational Databases (DDI Working Paper Series). http://dx.doi.org/10.3886/DDIOtherTopics02

Block, W.C., Andersen, C.B., Bontempo, D.E., Gregory, A., Howald, S., Kieweg, D., Radler, B.T. (2011). Documenting a Wider Variety of Data Using the Data Documentation Initiative 3.1, Longitudinal Best Practice, No. 1 (DDI Working Paper Series). http://dx.doi.org/10.3886/Longitudinal01

DDI Alliance (2014). DDI Moving Forward Project: Structure of DDI 4 and the process to create it. http://dditools.atlassian.net/wiki/display/DDI4/Structure+of+DDI+4

Edwards, M., Eisenhauer, J., Fry, J., Heus, P., Kolsrud, K., Moschner M., Nakao R., Thomas, W., (2009). Versioning and Publication, Best Practice, No. 8 (DDI Working Paper Series). http://dx.doi.org/10.3886/DDIBestPractices08

Gregory, A., Heus, P., Ryssevik, J. (2009). Metadata (RatSWD Working Paper 57).

Kramer, S., Leahey, A., Southall, H., Vompras, J., Wackerow, J. (2012). Using RDF to Describe and Link Social Science Data to Related Resources on the Web: Leveraging the Data Documentation Initiative (DDI) Model (DDI Working Paper Series). http://dx.doi.org/10.3886/DDISemanticWeb01

Lane, J. (2010). Linking administrative and survey data, in: Marsden P., Wright J., Handbook of Survey Research, 659-680.

Pienta, A.M., Alter, G.C., Lyle, J.A. (2010). The Enduring Value of Social Science Research: The Use and Reuse of Primary Research Data. http://deepblue.lib.umich.edu/handle/2027.42/78307

Vardigan, M., Heus, P., Thomas, W. (2008) Data Documentation Initiative: Toward a Standard for the Social Sciences, in: The International Journal of Digital Curation, Issue 1, Volume 3, 107-113.

Notes

1. Alerk Amin, RAND Cooperation, www.rand.org
2. Ingo Barkow, University for Applied Sciences Eastern Switzerland (HTW Chur), Switzerland; contact: ingo.barkow@htwchur.ch
3. Stefan Kramer, American University, Washington, DC
4. David Schiller, Research Data Centre (FDZ) of the German Federal Employment Agency (BA) at the Institute for Employment Research (IAB)
5. Jeremy Williams, Cornell Institute for Social and Economic Research
6. Data Documentation Initiative (http://www.ddialliance.org/)
7. http://www.ddialliance.org/alliance
8. http://www.w3.org/XML/
9. For more information on DDI see http://www.ddialliance.org/resources
10. http://basex.org/
11. http://exist.sourceforge.net/
12. http://www.mendeley.com/catalog/providing-olap-online-analytical-processing-useranalysts-it-mandate/
13. http://www.icpsr.umich.edu/icpsrweb/ICPSR/
14. http://msdn.microsoft.com/en-us/library/ms189887.aspx
15. http://www.oracle.com/technetwork/database/features/xmldb/index.html
16. http://www-01.ibm.com/software/data/db2/xml
17. http://www.ddialliance.org/community/listserv