

A Semantic Web based Identification Mechanism for Databases

Cristian Pérez de Laborda & Stefan Conrad
Institute of Computer Science
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Germany
Email: {perezdel,conrad}@cs.uni-duesseldorf.de

Abstract

In this paper we suggest the novel URI scheme `db` for identifying not only databases, but also their schema and data components like tables or columns. One of the features of this scheme is that it may not only be used for relational database systems, but for virtually any type of database or data source. We therefore have combined the advantages of both global uniqueness of URIs and the high flexibility of knowledge representation with RDF as part of the Semantic Web. With this novel identifier we are now able to enhance every data record exchanged between databases with metadata: an exact and identifying location of that data in the data source. As a result not only the system administrator is able to backtrack the data to its exact position in the data source but also the database system itself.

1 Motivation

Nowadays data is not exchanged and integrated manually anymore, but semiautomatically. Data exchanges between data sources run on their own after being set up once by a database engineer. Afterwards he only has to interfere if an error occurs or changes have to be done. In the former case the source of the data which produced the error has to be backtracked, so the error can be reconstructed and solved. This seems quite easy in an environment with two databases involved, but in more realistic environments we usually have quite more databases engaged in a data exchange. Since the data exchange formats and the data exchanged may only differ marginally, assigning the data to its corresponding sources may be quite complicated or even impossible.

Following the vision of completely autonomous databases exchanging not only their data, but also their metadata [3, 8], we will soon face a similar problem. In these cases, not only the data, but also the schema components have to be backtracked exactly to their sources. If the data source cannot be identified unambiguously out of the data exchange process, special identifiers for the data sources are needed.

Data sources do not only have to be identified in ongoing data exchanges, but also for later analysis. The information where a data record came from could be crucial

especially if errors do not appear immediately. If, in the meantime, the same data has been bidirectionally exchanged with multiple other data peers, the actual source of that item is not traceable any longer.

We have seen that the importance of identifying data sources is increasing with the rise of automation in data exchanges. In many cases internally assigned identifiers for the data sources within every peer would be enough. However if we start having autonomously and automatically acting databases exchanging data and metadata, this internal identifiers will not suffice any more. We can illustrate this with a small example.

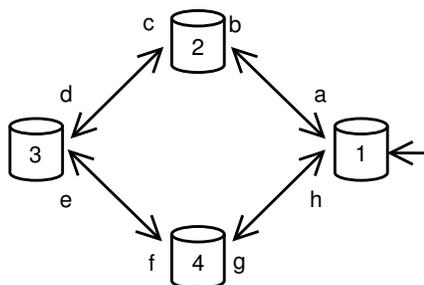


Figure 1: Multi-Peer-to-Multi-Peer data exchange

A data exchange system consisting of four databases (Figure 1) where each database synchronizes its data with two of the other databases. Every database propagates all data changes to both exchange partners. Each database has assigned an internal name for every exchange partner. Database 2 (db2) knows database 1 (db1) by the name *a*, db1 knows db2 by *b*, etc.

A data update on db1 is propagated to db2 and database 4 (db4). The first recognizes the data as coming from database *a*, for db4 the data comes from *h*. After it has been imported into the respective databases, the data records have to be propagated again. For this reason database 3 (db3) will get data from db2 and db4. Even if there was an identifier of the original source attached to that data, db3 would never be able to recognize the data as coming from one and the same source. The reason lies in the different names (*a* and *h*) for the source database. db3 is not able to recognize these two sources as being identical.

Apparently it makes sense to have a global identifier for the databases. This should not only identify the database itself, but also its components. The purpose of this paper is to propose such a global identifier.

2 Challenges designing an identifier

The first step in designing an identifier for databases is to analyze existing standards, if they may be suited accordingly. Obviously we would think at first of the *Abstract Syntax Notation One* (ASN.1) [11] or the *International Code Designator* (ICD) [10], but they do not fulfil our needs. ASN.1, a standard for the identification of all kinds of objects, has three main disadvantages. First, every company or institution interested in the identification of their database objects would have to register a subset of the

ASN.1 number space, which would arise additional costs. Second, the evolution of semi-structured data to XML and the high popularity of domain names instead of ip-addresses have shown that a data or address-exchange has to be done in a human readable and understandable manner. Using ASN.1 with its predominating representation *Object Identifiers* may only be understood with the aid of manually maintained mapping-tables. And finally there is no way of syntax validation included in the ASN.1 standard [16]. This is especially important for autonomous and automatic data transfers and identifications. Especially the possibility to validate the identifiers would give us an important instrument en route to an automated error-backtracking. Since ICD is a subset of ASN.1, the disadvantages mentioned above also apply. In addition, the number of possible enterprisers with assigned ICD numbers is limited to 9000 [10], an tremendous undersized amount for potentially identifying databases of every company or institution in the world.

Apart from the arguments above, ASN.1 and ICD identifiers persist only of a global identification number similar to URNs [17], but do not include a locator. Thus databases, trying to backtrack a data's source, would not be able to locate it, nor to contact it. For this reason and especially if we have autonomously acting databases, the location of a data source is an indispensable information, hence, it should be included in the identifier.

The most important feature of the identification mechanism must be flexibility. It has to suit to relational and object-oriented databases as well as to directories, legacy systems, or database types not yet developed. Additionally we have to handle with heterogeneity within the different database types themselves [19, 13]. A relational database server from vendor *a* may consist of different databases in which every database user has an assigned table space. Another relational database (even from the same vendor) may skip the database-user hierarchical level, so that one big table space is shared by all users. This sort of heterogeneity has to be considered as well.

Of course we could propose to introduce a new registration authority for world-wide unique database identifiers, but this center needs to be established first, the service would certainly not be free of charge, and it would have to be accepted by the community.

Concluding we need a flexible identifier for databases and their components without having a registration authority.

We have chosen to combine URIs [1] and the Semantic Web [2, 6] with its principle technology RDF [14] for creating our novel identifier. The result is an identifier as accepted as URIs are and as flexible as RDF is. The combination of both URIs and RDF allows us to create a flexible identifier with a virtually exhaustless address space, which is not only human readable but also machine understandable. All these features are possible without introducing a central registry.

3 Model

As mentioned above, we propose a novel model for identifying databases and their components combining the advantages of both **Uniform Resource Identifiers (URI)** and the **Semantic Web**.

An URI is a "compact string of characters for identifying an abstract or physical resource" [1]. It is predestinated for creating a global identifier for databases and their components. Presently there are several standardized URI schemes, for instance those for telephone numbers, email addresses, or host specific file names [12]. The best-known and most used URI scheme `http` stands for the **Hypertext Transfer Protocol** [7]. All URI schemes were introduced to create a global identifier for specific resources, which is also the aim of this paper. As there is still no appropriate URI scheme for databases, we introduce `db`, a novel URI scheme for databases. It identifies databases and their components, no matter of what type the database is and how that database is composed.

Every database involved into a data exchange can be reached with its unique IP address or Domain. We are thus able to use this address for the identification of the database server itself as the first part of our novel identifier. Additionally, we want to identify its components, for example its tables or columns which can usually be done through a kind of hierarchy. Since we cannot give a global hierarchy for all sorts of databases, the URI scheme itself has to be hierarchy independent. We have decided to use the **Resource Description Framework (RDF)** [14] of the **Semantic Web** for the identification of the single components. The advantage of RDF is definitely its flexibility. It gives us the possibility to create an arbitrary chain of attribute/value pairs for the unambiguous identification of the components wanted. A database column named `WorkerPK` could be identified by `Database=admin and Table=Worker and Column=WorkerPK` or just by `Column=WorkerPK` depending on the uniqueness of that column. As usual in RDF we have to create our own pool of attributes [5] and refer to them using namespaces [4]. In addition RDF gives this chain of attribute/value pairs a meaning, i.e. we are able to represent knowledge. With aid of that knowledge representation an exchange partner would know exactly that `dbs:Table` represents the construct *relation* known from relational model.

Herewith we are now able to identify a database server (with its IP address or Domain name) and all its components (with RDF). Unfortunately, there are still some challenges to be considered:

1. how to represent the RDF-Syntax [15] within the URI,
2. how to use internationalization inside the URI,
3. how to specify the corresponding namespaces,
4. how to address database servers located in private networks, and
5. how to incorporate temporal aspects,
6. how to deal with databases constantly changing its ip-address or not even having a real one

Challenges one, two, and three are solved below, the fourth, fifth, and sixth are subject to further research.

The most evident syntax for a database URI composed of an IP address and RDF attribute/value pairs would certainly be `db://ipaddress/prefix:attribute=value`. Unfortunately this is not allowed, because the colon is a reserved character [1] and thus must not be used. Instead we use the colon's ASCII representation `%3A`. As a result we get `prefix%3Aattribute=value` for the attribute/value pair.

A similar problem arises with non-standard ASCII characters. We cannot assume all the tables' or columns' names be restricted to the limited amount of ASCII characters, especially if we have to handle with data exchange crossing different language regions. In the RFC for the URI Generic Syntax [1] this restriction is virtually made, since it is not mentioned how to deal with non-standard ASCII characters. However we have a big advantage, since our URI will always appear within a data exchange, where an encoding has been arranged for the normal data. We just define the URI's encoding being the same as the data's. Was the encoding of the data exchange for instance ISO-8859-1 we would be able to have `prefix%3Aattribute=küche` as part of a valid URI. With the encoding ISO-8859-5 a valid URI could contain `prefix%3Aattribute=кухня`. An analogous example can be given for virtually every alphabet.

Besides the encoding we have to define the namespaces before using them in the URI. A definition within the URI would be a huge overhead, especially if the same namespace is used within several URIs. We thus assume as well that all namespaces have been defined before using them. For instance we are only allowed to use the prefix `dfs` if it has been defined previously as being the representative e.g. for `http://www.laborda.org/RDF/rdf-schema#`.

Figure 2 shows the syntax in EBNF of our database identifier. Please keep in mind, that the syntax is only given for US-ASCII characters and we have to ensure additionally that

- the corresponding encoding and
- all namespaces used

have been defined/set previously.

```

dburi      = scheme '://' hostport ['/' [attributes]]
scheme     = 'db'
hostport   = <hostport from RFC 1738>
attributes = attr *['&' attributes]
attr       = prefix '%3A' attribute '=' value
prefix     = alphanum
attribute  = alphanum
value      = alphanum
alphanum   = alpha | digit
alpha      = <alpha from RFC 1738>
digit      = <digit from RFC 1738>

```

Figure 2: EBNF Notation of db URI scheme

4 Example

After having defined our URI scheme for databases we will now give a short example. We therefore imagine a scenario in which two databases exchange data in an XML format. After a data record has been added to the database with the domain name `foo.de`, it has to be transmitted. The corresponding schema is described in Figure 3. The “has” relationship between the two entities is represented by an additional column `PersonFK` in the `Address` table with a foreign key relationship to the `PersonPK` column.

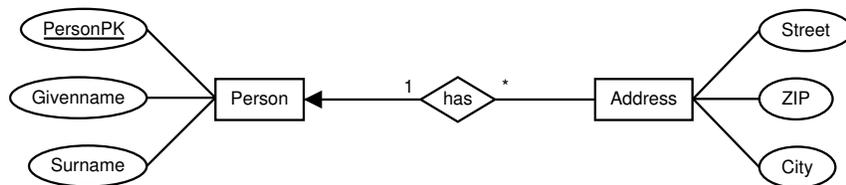


Figure 3: E/R representation of a database

The data which would be exchanged without our novel URI is given in Figure 4. Please note that the names of the tags do not always match the table or column names, thus they give no information where the data was originally located.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<mydatabase>
  <Employee>
    <PersonPK>123</PersonPK>
    <Givenname>John</column>
    <Surname>Public</Surname>
  </Employee>
  <Domicile>
    <PersonFK>123</PersonFK>
    <Address>Lane 12</Address>
    <ZIP>40225</ZIP>
    <City>Samplecity</City>
  </Domicile>
</mydatabase>
```

Figure 4: XML data exchange document

Using our novel URI scheme `db` we are now able to incorporate metadata into that exchange file (Figure 5). Every data record has metadata attached giving the information where that data came from exactly. This information does not only contain the address of the database server but also the exact position inside the database. With this information every exchange partner can understand the metadata and is therewith able to determine the exact position of every item received in the data source.

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<mydatabase location="db://foo.de/dbs%3ADatabase=admin" xmlns:dbs="http://www.laborda.org/RDF/rd-schema#">
  <Employee location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Person">
    <PersonPK location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Person&dbs%3AColumn=PersonPK">123</PersonPK>
    <Givenname location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Person&dbs%3AColumn=Givenname">John</column>
    <Surname location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Person&dbs%3AColumn=Surname">Public</Surname>
  </Employee>
  <Domicile location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Address">
    <PersonFK location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Address&dbs%3AColumn=PersonFK">123</PersonFK>
    <Address location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Address&dbs%3AColumn=Street">Lane 12</Address>
    <ZIP location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Address&dbs%3AColumn=ZIP">40225</ZIP>
    <City location="db://foo.de/dbs%3ADatabase=admin&dbs%3ATable=Address&dbs%3AColumn=City">Samplecity</City>
  </Domicile>
</mydatabase>

```

Figure 5: XML data exchange document with our novel URI

5 Conclusion and Future Work

In this paper we have suggested a novel URI scheme for identifying and locating not only databases themselves, but also their schema and data components. This scheme guarantees a global uniqueness of the identified components and remains as flexible as RDF is. Herewith we are not only able to identify relational or object-oriented databases but also legacy or file systems.

The `db` scheme consists of two parts. One is the locator of the database (i.e. its IP address) and the second part is a locator inside the database system, realized with RDF. This fact ensures, that we do not only write attribute/value pairs but represent knowledge of the database's structure. Herewith the exchange partner, being a human or a computer, is instantly able to read and understand the data.

There are still some items to be solved in future work. The most important task is to incorporate a temporal aspect [18] to the URI scheme. Herewith we would be able to identify the data source not only at the time the data record was created, but indefinitely. A reasonable solution could be introducing a mandatory attribute giving the time when the data record was created, e.g. `foo:recordCreationTime=1057159782` using UNIX time or `foo:recordCreationTime=2003-07-02T152942GMT` using one of the numerous ISO 8601 [9] formats.

A further challenging problem concerns databases with changing or not reachable ip-addresses, e.g. mobile databases or those in private networks. They either change frequently their address, or do not have an unambiguous one. This has to be evaded probably by introducing artificial addresses or names. Please note, that this fact does only affect the first part of the URI scheme and not our novel method of object identification inside the database.

References

- [1] Tim Berners-Lee, Roy Thomas Fielding, and Larry Masinter. RFC 2396: Uniform Resource Identifiers (URI): Generic syntax, 1998.
- [2] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.
- [3] Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John My-

- lopoulos, Luciano Serafini, and Ilya Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Workshop on the Web and Databases, WebDB*, 2002.
- [4] Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML — World Wide Web Consortium Recommendation. <http://www.w3.org/TR/REC-xml-names>, 1999.
- [5] Dan Brickley, Ramanathan V. Guha, and Brian McBride. RDF Vocabulary Description Language 1.0: RDF Schema. <http://www.w3.org/TR/2002/WD-rdf-schema-20021112/>, November 2002.
- [6] Edd Dumbill. The Semantic Web: A Primer. <http://www.xml.com/pub/a/2000/11/01/semanticweb/>, November 2001.
- [7] Roy Thomas Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. RFC 2616:Hypertext Transfer Protocol - HTTP/1.1, 1999.
- [8] Alon Halevy, Zachary Ives, Dan Suciu, and Igor Tatorinov. Schema Mediation in Peer Data Management Systems. In *19th International Conference on Data Engineering*, March 2003.
- [9] International Organization for Standardization. *ISO 8601:1988. Data elements and interchange formats — Information interchange — Representation of dates and times*. International Organization for Standardization, Geneva, Switzerland, 1988. See also 1-page correction, ISO 8601:1988/Cor 1:1991.
- [10] International Organization for Standardization. ISO/IEC 6523-1:1998 Information technology – Structure for the identification of organizations and organization parts – Part 1: Identification of organization identification schemes, 1998.
- [11] International Telecommunication Union. Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T Recommendation X.680, 2002.
- [12] Internet Assigned Numbers Authority. Uniform Resource Identifier (URI) SCHEMES. <http://www.iana.org/assignments/uri-schemes>, 2003.
- [13] Won Kim, Injun Choi, Sunit Gala, and Mark Scheevel. On Resolving Schematic Heterogeneity in Multidatabase Systems. In W. Kim, editor, *Modern Database Systems*, chapter 26, pages 521–550. ACM Press, New York, NY, 1995.
- [14] Ora Lassila. Introduction to RDF Metadata. <http://www.w3.org/TR/NOTE-rdf-simple-intro-971113.html>, 1997.
- [15] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>, February 1999.
- [16] Michael Mealling. RFC 3001: A URN Namespace of Object Identifiers, 2000.
- [17] Ryan Moats. RFC 2141: URN Syntax, 1997.
- [18] Stefano Spaccapietra, Christine Parent, and Esteban Zimányi. Modeling time from a conceptual perspective. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management, Bethesda, Maryland, USA, 1998*, pages 432–440. ACM, 1998.
- [19] Pepijn R. S. Visser, Dean M. Jones, Trevor J. M. Bench-Capon, and Michael J. R. Shave. An Analysis of Ontological Mismatches: Heterogeneity versus Interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA, 1997.