What's in a Federation?
Extending Data Dictionaries with Knowledge Representation Techniques

Wolfgang Benn
Chemnitz University of Technology • Management of Data
P.O. Box 964 • D-09009 Chemnitz
benn@informatik.tu-chemnitz.de

1. Introduction

Databases and knowledge representation languages have a rather different view upon data: knowledge representation languages describe a universe of discourse in a taxonomy and allow a user to ask epistemic questions against the relationships between concepts and roles. However, no data structures, data locations, nor any information about the existence or availability of data can be found in a taxonomy -- even not if it includes an assertion that describes a particular data item.

Relational databases provide users with schemata. Schemata describe in detail the data structures of sets of persistent data items. Data dictionaries, included in these systems, tell about data existence and its availability. Anyway, these tools do not provide the entity view, relationships between entities are merely implicit, and no question about the universe of discourse that is behind a schema will get an answer.

Object-oriented databases provide users with class hierarchies as schemata. They support the entity view -- is-a as well as part-of relationships are explicit. Nevertheless, an information about the universe of discourse is not given as well.

In a federation of systems -- databases and applications, for instance -- the situation gets worse. Databases may be heterogeneous in their modeling technique: some will follow the object-oriented the majority certainly follows the relational paradigm. How does a user get to know what data is available in a federation, if he wants to build a new application? How does that user get to know how he may access a particular data item? How does he know that the selected data item is semantically correct concerning the context of his application?

If he can access a federated data dictionary, it will provide him with the technical information about the data in a common data model -- similar to the global conceptual schema of a distributed database. If such a tool does not exist, the user must read all available schemata from all available federation components (i.e., he must know about all languages, data models, and dialects that the local components of the federation individually use).

In the remainder of this paper we will briefly introduce a module that coordinates a federation of systems and that hosts a central data dictionary. It is the module, which we will extend to provide users with an entity view upon the information available in a federation.
We introduce the logical architecture of a prototypical implementation of this module in section 2 and describe some extensions that we made in section 3. In section 4 we specify some ideas of the mentioned extension, conclude in section 5 and give some literature in section 6.

2. The Federal System Manager

The Federal System Manager (FSM) is a module that coordinates a federation of autonomous systems. These systems can be applications or services like databases, which may link to the FSM to form a federation for some particular tasks. Afterwards they can leave the federation and run again as autonomous systems. This idea is rather similar to the concept of multi-agent systems.

The FSM performs a minimum of three tasks: The first one is to run a protocol that enables the linkage process and guarantees a negotiation of autonomy aspects to the components, if these want to join or leave the federation. Second, the FSM must provide a uniform view upon all information that is available to applications of the federation through a so-called Common Data Model (CDM). Third, it must support an exchange of information, i.e., data types and data itself, between members of the federation. We will detail these tasks and concentrate on the second one.

Comparing an FSM with the Common Object Request Broker Architecture (CORBA) [1] the FSM is an object broker that looks at databases as service providing objects and applications as clients that request these services. Commonly known services from database components are storage, retrieval, update, etc.

Moreover, the FSM is an object itself! It provides services like data and type exchange. It contains a Federal Data Dictionary (FDD) that allows a user to retrieve the information contents of the actual federation under several aspects. It is our aim to extend this Federal Data Dictionary with knowledge representation techniques to better support users in their retrieval than before.

2.1. The FSM Prototype

The currently implemented FSM prototype has its roots in an ESPRIT project, finished in 1991 [2,3,4,5,6]. The prototype mainly follows the reference architecture for interoperable systems given in [7] and includes a repository according to the Information Resource Dictionary Standard IRDS [8].

This standard defines a four-layer architecture with (top down)

- a meta-meta layer that describes the model of the meta layer descriptions -- which is in our case the Common Data Model of the FSM, a frame work that basis on the Abstract Data Type (ADT) idea --,
- a meta layer where we find the description of schemata -- which is in our case a description of the federation components data models --,
- a schema layer where the data descriptions are located -- which is in our case the data types that are defined in schemata of databases or in type declarations of applications --, and
- an application data layer where we finally find the application data itself.

The Meta-Meta Layer

To enable the description of schema descriptions we implemented a common data model.

In the literature we found many different approaches to implement a CDM -- the approach most often used, however, was the object-oriented. Thus, we asked ourselves, what is the kernel idea of the object-oriented paradigm that makes it suitable for a CDM. We found out that it probably is the idea of Abstract Data Types.

Thus, we implemented a frame work, which is actually not a real data model but a tool box [2]. It allows a user to describe the structure and semantics of those elements, which he uses to describe a schema, similar to the ADT concept (see next paragraph).

The CDM that we implemented is very similar to the Interface Description Language (IDL) of the CORBA specification [1] -- because its purposes are rather similar. IDL is a language, which describes object services in an intermediate way and the CDM describes entities (application objects) in an intermediate way.

An IDL description is mapped into a real programming language and the object services are available for all programs written in this programming language. Application objects described in our CDM are (under certain conditions) transformable into all data models that are represented in the FSM.

The Meta Layer

An extension of the IRD standard was made for the meta layer. If the FSM supports an exchange of data between components, it must be able to transform data between the different individual data descriptions. These descriptions follow type or schema declarations, which use data model elements. Thus, our meta layer has to include a suitable sub-set of the component data model for each involved component. Moreover, it must include some rules that guide the transformation of entities between these data model sub-sets.

However, the description of a data model sub-set is somewhat more complex than the description of a schema. While a schema merely consists of data structures, a data model usually includes data types and data type semantics. The meta layer of our FSM includes both (the assignment of a set of operations to a data type that makes up the type's semantics in the data model of a component is currently under implementation).

To enable the exchange of data and schema information between components the system administrator of each federation component defines the relevant structural part of his component data model types with the CDM types and assigns some procedures that make up the semantics of these data types. He inserts the necessary data model knowledge into the meta layer using the meta-meta layer elements.

For instance, from an object oriented data model the administrator defines the structural parts of the concept CLASS and assigns at least one particular routine that performs inheritance similar to his individual data model.

This information is provided through an interface, which is the so-called Data-Model-Profile. It is an ASCII file with a particular syntax that is parsed. Then the information is kept in a knowledge base -- the FSM Meta Knowledge Base.

The Schema Layer

Databases, as components of a federation, use database schemata. Applications use data type definitions to declare their application types.

The FSM reads these schemata and declarations and interprets the used data types through the information of the meta layer. Application entities are transformed into entities of the CDM and then -- for storage purposes -- transformed into entities of a database data model.

The entity information in CDM-format is stored in the Federal Data Dictionary (FDD) for retrieval purposes.

The Application Layer

Finally the data that comes from applications is stored in databases that have joined the federation, that are represented through meta-information in the Meta Knowledge Base, and that are willing to perform the storage process after a negotiation of their autonomy rights.

Of course, the data is not stored as CDM-typed data but is typed according to the data model of the involved database system. The interpretation of binary data runs the same way as the transformation of type information: It goes from the data model of the application towards the CDM and from the CDM to the database data model, and vv.

3. Extensions of the FSM Prototype

Since 1991 the FSM prototype has been completed by some student's work.

The Federal Data Dictionary of the prototype contained information about data type declarations, the types of application entities, and the structure of these entities -- as well, access rights were included. It did not include any technical information about the availability of entities or schemata.

We extended the FDD and it now contains technical information about the federation components. The meta layer includes information about the technical system that hosts the application or the database system. The schema layer includes information about the technical availability of entities [9].

The lack of a docking mechanism and a protocol to negotiate autonomy was another problem of the original FSM prototype. It was a static system with two applications, a database system and the FSM with hard wired mechanisms to read data type declarations -- database schemata could not be read, nor was it possible to link another database system with the FSM.

Now we have implemented a link mechanism that generalizes the old one [10]. We now use a FSM-Bind module that binds a component -- either a database system or an application -- if it includes our FSM-Bind-Agent.

The FSM-Bind-Agent acts as a client to the FSM-Bind module, which is the server, and performs the link process between FSM and component. It runs an implemented protocol for start-up and shut-down situations and uses the Remote Procedure Call (RPC) technique.

After linkage the FSM-Bind-Agent passes control to a so-called FSM-Agent, which performs the information exchange and the retrieval of schema information via the Remote Data Access (RDA) protocol.

What is still missing, is a user friendly retrieval facility that completes the Federal Data Dictionary. We will describe our ideas in the next section.

3.1. Extensions of the FDD

Data dictionaries offer technical information to users -- and exactly this can be expected from our Federal Data Dictionary as it is currently implemented. If a user wants to build a new application he looks into the FDD and looks up some data structures that he wants to re-use. Then he includes the chosen data structures into his new schema (the FSM provides some commands to do so) and runs his application.

This user is unable to check whether his new schema violates the semantic integrity of the universe of discourse of the actual federation because he can not ask the FDD to present him semantic relations between entities.

We wish to provide such a user with an extended Federal Data Dictionary, which shows the contents of a federation from various levels of abstraction. If this extended data dictionary has a graphic interface the user will use a mouse to easily request the change of levels. Which are these levels?

Taxonomy Level

The highest level presented, should be a taxonomy upon the universe of discourse. It could be the union of all schemata (and may be data type declarations of applications) of local database components, which we previously transformed into the abstraction level of a concept language. This level would represent the data of a particular federation without any technical details. Here the user could look-up the real-world context of an entity and might ask questions about the relationships between entities. It is the level that KL-ONE like languages usually offer to users with their T-Box.

Concept Languages separate between the terminological (T-Box) and assertion knowledge (A-Box). The task, which we have to perform is to abstract the technical information from schemata and data type declarations to concepts of concept languages. In [11] we find a theoretical basis that allows us to express database schemata with concept languages.

Moreover, the authors show that classification is then available for entities of schemata -- and we found out that the implementation of a classificator is surprisingly supported through an algorithm, which we use within the FSM to detect data type intersections for types from different data models. This algorithm follows perfectly the above mentioned steps for a classification of concepts.

Anyway, if we make the is-a and part-of relations of entities from schemata explicit and suppress the technical information, then we can ask questions against a schema similar to the questions against a taxonomy.

The implementation of this level may use intermediate language representations that follow the idea of attributed trees. This model allows us to determine the degree of entity detail information, which we want to present, by cutting the tree at a certain level. The information above the cut is presented as concept. The rest is hidden until requests from other levels of our retrieval interface force it to become visible.

Apparently, we address some open questions if we want to extend a data dictionary with knowledge representation features:

How do we find a way to reconstruct the entity view from relational schemata with normalized relations? Any automatic evaluation of foreign keys -- which is the only data model construct that can be used to express sub-part relationships, set-inclusions, and entity-inclusions within the relational data model -- finally depends on the support of a human. A machine may solely hypothesize is-a relations between entities. Thus, our entity re-constructor can not be a completely automatic component. It has to include a dialogue component to keep in touch with a human expert, but it may be a component that is able to learn.

Schema Level

On a second level, the schema level, in a detailed view, the user should have access to the more technical details of entities and should see what attributes an entity make up, where the information resides within the federation, whether and when it is accessible for him.

This level is comparable with an extended Entity-Relationship level where we added attributes about data distribution and data availability to the usual representation of entities, attributes, and relationships.

We realize this view by an FDD retrieval, because our directory includes the structure information of entities in a neutral representation and the information about the availability of these entities.

Syntax Level

Finally, the user may get what he always got from databases: the pure schema information. If he asks for this, he will get an excerpt of a schema of one or more particular local components of the federation -- and he should decide himself whether he would like to receive this information in the format of a common data model or in the individual format of the involved local federation components.

4. First Steps toward the Taxonomy Level

Concerning the integration of abstract schema representations into one taxonomy we did some work in advance and evaluated an idea, published in [12]. It proposed the assignment of fuzzy values to relationships to determine the is-a of an entity.

We took this idea and tried to use probability values for the integration of different schemata into one -- to simulate the situation that comes up if we have to integrate abstracted schemata from components into one taxonomy. It was a first guess to cope with modeling heterogeneity.

The basic assumption behind our tests was, that the insert of knowledge into a taxonomy is an evolutionary process and that we ask "is B a A *or* a C" and not "how probably is B a A *and* a C".

We defined a value C_T (E_i, E_j) for the correctness of a is-a relationship between two entities E_i and E_j in a taxonomy for the federation. Such a value is assumed to be assigned to each is-a relationship within that taxonomy. Similar to C_T we defined a C_S (E_i, E_j) as a value for the correctness of a is-a relationship in a local schema.

Next we said that S_T (E_n) and S_S (E_n) are the sets of all super-concepts of a concept in the taxonomy and an entity in a local schema.

Finally, we defined two functions, which were necessary to calculate the probability values during the integration process.

The first function was called INIT and initialized an initial taxonomy with the value 1 for all is-a relationships: C_T $(E_i, E_j) := 1$.

The second function included a case statement and was called CALC. It calculated the initialized values according to the new schema. The first case, C_1, was used if a relationship was found in a schema -- it corresponds with the INIT function for the taxonomy -- and set $C_S (E_i, E_j) := 1$. We assume that the designer of the schema did a good and correct work.

The second case, C_2, was used, if we find a relationship within the schema but not within the taxonomy. We insert the relationship into the taxonomy and give it the value $C_T (E_i,E_j) := C_S (E_iE_j) \div \text{card} (S_T (E_i) \approx S_S (E_i))$.

This approach seems to be correct because we can not guarantee that the taxonomy was correctly initialized with relationships. Moreover, an insertion of a new relationship affects the probability value of another one because there must be a reason why a particular application domain needs this new relationship. It may be, that the already existing relationships do not have the importance, which we have expected.

Finally there is the case C_3. In this case we see a relationship within the taxonomy but miss it in a schema. We interpret that relationship as "possible but unnecessary" within this application domain and "insert" it into the schema with $C_S (E_i,E_j) := C_T (E_i,E_j) \div \text{card} (S_T (E_i))$.

Then we made three assumptions:
a) The increase of probability of one particular relationship is given by its existence in schemata and causes a decrease of probability for those relationships, which are often missed.
b) The results of calculations about the overall probability for a particular relationship is included into the taxonomy.
c) Results are calculated through the geometrical mean of the two probability values from the taxonomy and from a schema.

With these assumptions and formulas we tested the integration of six schemata into a taxonomy, which was initialized with one relationship "B is-a A". Four of these schemata included the relationship "B is-a A" (we call them the A-type schemata). Two included "B is-a C" and not "B is-a A" (we call these the C-type schemata).

In a first test, we inserted a C-type schema first and afterwards both relationships had the same value (0.71) in the taxonomy. A four-times insert of the A-type schemata brought the value of the "B is-a A" relationship up to 0.98 and the value of "B is-a C" fell down to 0.18 -- similar to the predicate "insignificant" or "incorrect". A final insert of a C-type schema, however, gave a new balance to both values, which was 0.69 for the "B is-a A" and 0.42 for the "B is-a C" relationship.

A second test gave surprising results: We inserted the two C-type schemata and then four times the A-type schemata. This gave a high value to the "B is-a C" relationship first -- the balance was 0.5 for "B is-a A" and 0.84 for "B is-a C" -- and a final value of 0.96 for "B is-a A" and 0.37 for "B is-a C".

While the first test showed that the late insert of an apparently insignificant relationship makes the value system unstable, the second test showed that an early insert of the two C-type schemata prevents the alternative relationship to fall down to an "insignificant" valuation.

Anyway, both value calculations were highly sequence dependent, and we suspected the second assumption as the reason for it. Thus we tried again without this assumption. We inserted into C_3 a variable: $V (E_i)$ counts the number of schemata without a particular relationship and the calculation C_3 changed to

$$C_S (E_i,E_j) := 1 \div (V (E_i) + 1).$$

This does not change much and we were stuck to the question: Is the insert of knowledge really an evolutionary process or is it correct to calculate probability values from the arithmetic mean of all values from schemata?

5. Conclusion

The proposed extended data dictionary gives a twofold benefit. At first, a user who wants to build a new schema for an application in a system federation can check which entities already exist, which of them he can re-use within his application, and which one he has to add or modify.
Second, an administrator can test the correctness of an existing schema against the universe of discourse. He can check the completeness of relations between entities by looking-up the taxonomy, where he would find the collection of all relationships between entities -- and eventually a probability value of the necessity or reliability of an individual relationship.

6. Literature

[1] *The Common Object Request Broker: Architecture and Specification*, OMG Document Number 91.12.1, Revision 1.1, Draft

[2] W. Benn, G. Junkermann, H. Kalweit, Ch. Kortenbreer, G. Schlageter, X. Wu: *The Conceptual Ob-*

ject Manager Document, University of Hagen, Computer Science Report Nº 99, 1990

[3] W. Benn, Ch. Kortenbreer, X. Wu: *Towards Interoperability: Vertical Integration of Languages with a KBMS*, GI-Fachtagung "Datenbanksysteme in Büro, Technik und Wissenschaft" (BTW 91), Springer-Verlag, 1991

[4] W. Benn: *KBMS Support for Multiple Paradigm Applications*, in [16]

[5] W. Benn: *KBMS Support for Conceptual Modeling in AI*, 3rd International Conference on Tools for Artificial Intelligence, 1991

[6] W. Benn, Ch. Kortenbreer, G. Schlageter, X. Wu: *On Interoperability for KBMS Applications - The Horizontal Integration Task* **-**, 8 th Intl. Conference on Data Engineering, Phoenix, AZ, 1992

[7] A.P. Sheth, J.A. Larson: *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*, ACM Computing Surveys (1990) 3

[8] DIN 66 313, *Rahmenangaben für Systeme zur Verwaltung von Informationsrecourcenverzeichnissen*, DIN Deutsches Institut für Normung e.V., Berlin, 1992 (same as ISO/IEC 10 027)

[9] J. Hunstock: *Erweiterung einer Wissensbasis zur Realisierung von universellem Polymorphismus in föderativen Systemen um technische Informationen autonomer Systemkomponenten (Extending the Meta-Knowledge Base of the FSM by technical information)*, thesis for diploma, Chemnitz University of Technology, 1993

[10] M. Schöne, S. Herold: *Konzeption und Implementierung eines Protokolls und zugehöriger Systemkomponenten zur Integration von Datenbanksystemen in einer Föderation (Design and implementation of a protocol for the integration of database components into a federation)*, thesis for diploma, Chemnitz University of Technology, 1994

[11] S. Bergamaschi, C. Sartori: *On taxonomic reasoning in conceptual design*, ACM TODS (1992) 3

[12] P. Fankhauser, M. Kracker, E. Neuhold: *Semantic vs. Structural Resemblance of Classes*, ACM SIGMOD Record 20 (1991) 4