Organising Knowledge of a Federated Database System to Support Multiple View Generation

D. D. Karunaratna W. A. Gray N. J. Fiddian

Department of Computer Science, University of Wales-Cardiff, UK
{scmddk,wag,njf}@cs.cf.ac.uk

Abstract

In this paper a knowledge base structure to support multiple view generation over a federation of loosely-coupled heterogeneous databases available over a computer network is described. In such a federation user information requirements change and evolve as their awareness of the contents of the federation improves. This means that users need different integrating views if they are to realise the full potential of the federation. To facilitate the creation of integrating views, users need to know what is available in the federation, how the available information is related semantically and which information has been used together previously. Once integrating views are created, users must be informed when their views become invalid due to changes in the participating databases. The knowledge base is structured primarily to assist users in such tasks. The knowledge stored in the knowledge base is obtained by analysing the meta-data of the databases of the federation and from their respective owners. The knowledge base is built in bottom-up fashion by extracting and merging incrementally, the meta-data of the DBs. The canonical data model used to organise both the meta-data and the knowledge base is also described. Finally a workbench of tools we have developed to assist users to create the knowledge base and to generate integrating views over the federation is outlined.

Proceedings of the 5th KRDB Workshop
Seattle, WA, 31-May-1998

(A. Borgida, V. Chaudhri, M. Staudt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/

1 Introduction

With the growth and widespread popularity of the Internet the number of databases available for public access is rapidly increasing both in number and size, while at the same time users and application programs are increasingly requiring access to data in an integrated form, from multiple pre-existing databases [CS91, RS94, SPD92]. These pre-existing databases are typically autonomous and located on heterogeneous hardware and software platforms. Different users have different needs for integrating data and their requirements may change over time as new databases become available. Hence the same database may participate in many different ways in multiple integration efforts.

A system that supports operations on multiple databases (possibly autonomous and heterogeneous) is usually referred to as a *multidatabase system* [BHP92, KS91, Lit88, PBE96]. There are two popular system architectures for multidatabases: tightly-coupled federation and loosely-coupled federation [CHS91, KS93, SL90]. In a tightly-coupled federation, data is accessed using a global schema(s) created and managed by a federated database administrator(s). In a loosely-coupled federation it is the user's responsibility to create and maintain the federation's integration regime. They may access data either by means of views defined over multiple databases [HM81, HM85] or by defining a query using a multidatabase language [Lit88, LMR90] which enables users to link concepts within queries.

The creation of an environment that permits the controlled sharing and exchange of information among multiple heterogeneous databases is identified as a key issue in future database research [HM93, SSU90]. We believe that a loosely-coupled architecture provides a more feasible environment to meet users' evolving and changing information requirements across the disparate databases available over the global informa-

tion infrastructure (Internet), as it naturally supports multiple views which are dynamic, and it also avoids the need to create a global integrated schema which makes evolution harder. It is also envisaged that the primary issue in the future will not be how to efficiently process data that is known to be relevant, but rather to determine which data is relevant for a given user requirement or application (resource discovery [MKSI96]) and to link this information together in different ways to meet user needs in a changing and evolving situation[MBP95, SK92, She91]. Success in resource discovery and building integrated views over heterogeneous databases depends crucially on an understanding of the information content and semantics of the schema components of the participating databases [DKW+91, EM97, SGN93]. A number of projects have been reported in the literature which capture semantics of schema components by using pre-existing Knowledge Bases (KBs) such as ontologies [GMI97, MKSI96], concept hierarchies [YSDK91] and semantic dictionaries [CA97]. We argue that linking schema components with ontologies will not provide a complete environment for global users to build integrated views and to keep those views consistent with local schema evolution and modifications. It also limits the scope of the component databases to the view of the compilers of the KB at the time of its creation. Thus it is essential to complement such an environment with static and dynamic knowledge about the participating databases (e.g. location, access permission, how frequently the data is updated), schemas (e.g. structure, representation aspects of schema objects) and existing user generated views to exploit the full potential of the federation. In this paper we describe a canonical data model and an architecture for a KB, based on that data model, which organises knowledge about a loosely-coupled federation of heterogeneous databases. The KB is aimed at assisting global users to build integrated views and helping them to keep those views consistent with local schema modifications, thus preserving local autonomy. It is not a static and pre-established KB but is created and evolved incrementally as databases join the federation and as views are generated over the federation by users.

The remainder of this paper is organised as follows: in section 2 we describe a canonical data model used to organise meta-data of the component databases in the federation and the knowledge of the knowledge base. The structure, content and the organisation of knowledge in the knowledge base is explained in section 3. In section 4 we outline a workbench of tools designed to assist users to create and maintain the knowledge base and to generate integrating views over the federation. Finally, section 5 presents conclusions.

2 The Canonical Data Model

Our KB is created and evolved incrementally in a bottom-up fashion by analysing meta-data of databases and eliciting knowledge from the database owners as they join the federation and when they evolve the databases. Hence our system requires meta-data of the participating DBs (component DBs) to be extracted, structured and stored separately based on a canonical model so that they can be analysed, enhanced and merged with the KB subsequently. The main aim of our canonical data model is to structure the meta-data of the component DBs to facilitate the creation of the KB so that the users can utilise the knowledge in the KB during different stages (e.g. resource discovery, identification of semantically related schema elements) of the integrating view creation process.

In DBMSs both meta-data and the extension of a DB are typically represented by means of the same data model and managed by the same DBMS. Since the main objective of DBMSs is to store and manage data, the data models on which they are based are not rich enough to store knowledge describing the content of the DBs managed to a wide user community. Neither are DBMSs equipped with functionalities to elicit, structure and manage such knowledge. Experience with early prototype systems also suggested that integrated data and meta-data management is difficult in many application areas as meta-data differs significantly from data [McC82]. During the past two decades, a number of semantic data models appeared in the literature aiming at capturing more and more semantics of the data stored in databases. These semantic data models are typically different from the data models used to store DB extensions in traditional DBMSs, necessitating separate repositories and functionalities to be designed to store and manage them if the existing DBs and DBMSs are to be used without major modifications. The types of meta-data to be captured for all possible applications cannot be fully anticipated, requiring the data models used for this purpose to be flexible enough to represent a wide variety of meta-data and to allow meta-data to be added incrementally.

The canonical data model we use to organise knowledge about a federation is an associative network of terms and is based on the ideas of Telos [MBJK90], WordNet [Mil95] and binary-relational storage structures [Fro82]. The terms correspond either to schema objects (e.g. relations, entities, classes) or to their attributes defined in the component DB schemas and represented as nodes in the network. Terms are related with other terms via binary relations which are represented as links between the corresponding nodes

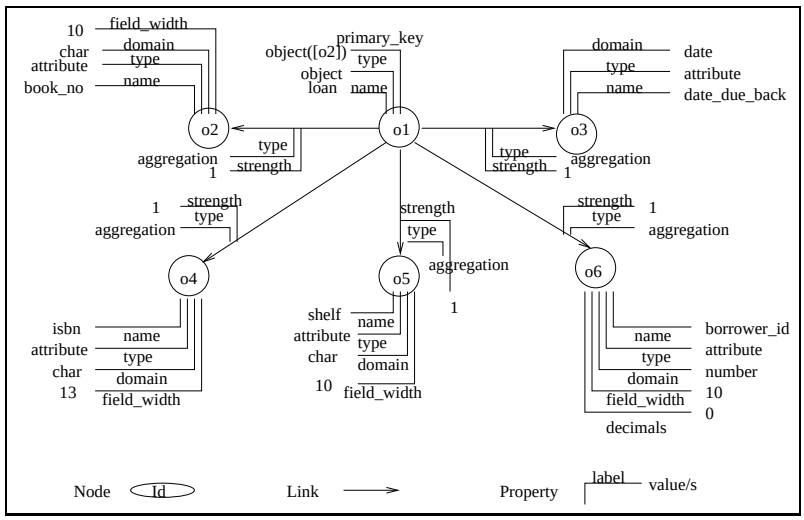

Figure 1: The schema structure built for part of a sample database table (relating to an object called 'loan')

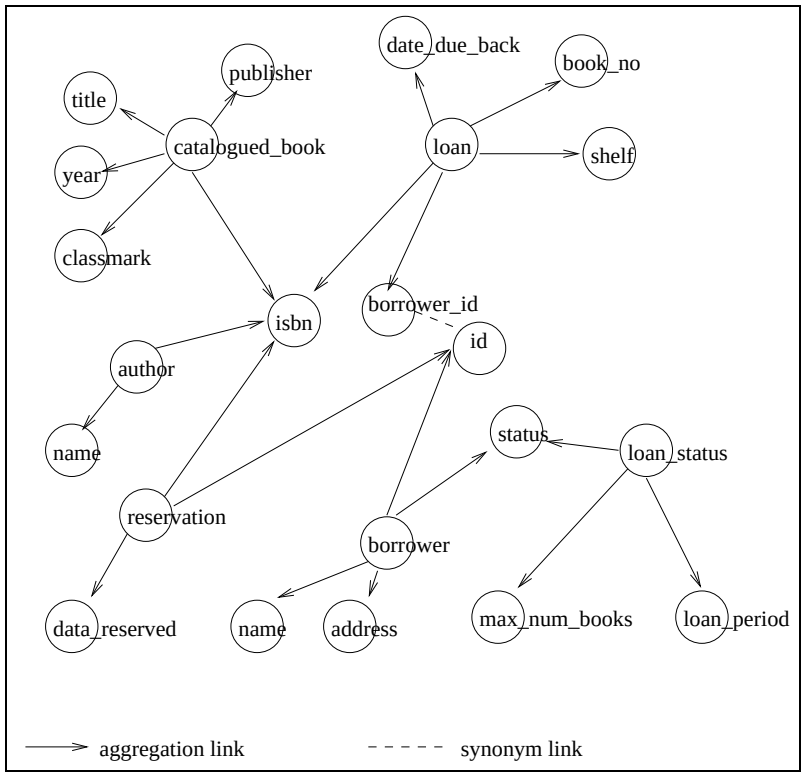

Figure 2: The schema structure created for the sample database

in the network. In the model both schema objects and their attributes are treated alike as full-fledged objects, allowing schema object attributes to have their own attributes. Each node and link in the network is given a unique system-generated internal identifier and is assigned a set of *properties* some of which are essential. A property is given a *label* and has a *value(s)*. The properties of a node (or a link) describe such things as the implementation aspects (e.g. object, attribute), data semantics [SM91], visualisation aspects (e.g. shape, size) etc. of the corresponding terms (or relations). The labels classify properties into types. The essential property types defined for the nodes and links are given below.

2.1 Essential property types of nodes

- **name :** describes the external name(s) of the corresponding term. A node may have multiple external names. The set of all external names attached to a node is analogous with *SynSet* in WordNet [Mil95].

- **type :** describes the type of schema element the corresponding term represents and can take a value from the set {object, attribute}.

2.2 Essential property types of links

- **type :** describes the type of the corresponding binary relationship, and can take a value from the set {aggregation, isA, positive-association, synonym}. These links are generated among nodes as below.

  - aggregation : links two terms T_1 and T_2 when T_1 corresponds to a schema object O_1 and T_2 corresponds to an attribute A_i of the object O_1.

  - isA : links terms T_1 and T_2 when T_1 and T_2 denote two schema objects O_1 and O_2, respectively, and a specialisation/generalisation relationship exists between them.

  - positive-association : links terms when an association other than *aggregation* and *isA* is defined or derived between objects in the schema. In a relational schema we use the foreign key definitions to derive positive-associations between schema objects.

  - synonym : links two differently named schema terms when they correspond to two schema object attributes that denote the same real-world aspect.

- **strength :** represents how closely the connected terms are related in the DB context and takes a value in the interval $(0, 1]$, where the value 1 indicates a definite relationship, a relationship explicitly defined among the corresponding schema objects. The links with strengths less than 1 are referred to as *tentative links*. One of the applications of tentative links is described in the following section.

The DB owners are expected to assign schema elements with meaningful names, to discover implicit relationships among schema objects as far as possible and to record them in the canonical data model prior to merging it with the KB. For example in relational DBs, the same attribute may exist with different names in different relations to create implicit relationships between relations. If the database management system (DBMS) supports foreign key constraints then such relationships can be stored in the schema explicitly. However there are relational DBMSs (e.g. INGRES) that do not support the specification of foreign key constraints through their Data Definition Languages (DDLs). Even in DBMSs where the foreign keys are supported, DB designers are not expected to define all possible foreign keys. Hence tools may be used to analyse both the schema and data to discover such relationships and to create tentative links among schema objects in the canonical model. The DB owners can inspect these tentative links and can either confirm (i.e. modify the strength to 1) or reject (i.e. delete the link) them.

The nodes and arcs of the network are represented uniformly in the KB as PROLOG predicates of the form $p(identifier, label, value{-}clause)$. The label and value-clause of a predicate describe a particular property and its value(s) of a node (or a link). Application programs may use different properties of the nodes for different tasks. For example a graphical user interface (GUI) accessing the KB may use only the properties that are required for visualisation to draw nodes and arcs on the screen. We are planning to extend the model to represent integrity constraints of the component DBs as PROLOG rules with a corresponding property to attach the rules to their respective nodes.

We refer to the meta-data of a component DB structured according to the canonical data model as a *schema structure* of the corresponding DB. An example schema structure, including properties of the nodes and arcs, generated for the ORACLE relation "loan" created by using the SQL commands given in the appendix is shown in Figure 1, while the complete schema structure, excluding node and arc properties, generated for the entire database is given in Figure 2. Note that in Figure 2 node ids are replaced with values

of the property labelled "name" of the corresponding nodes.

3 The KB

Our KB is structured into four layers : a *concept layer*, a *view layer*, a *meta-data layer* and a *DB layer* (Figure 3), providing information for different stages in the view generation process. The information in each layer is organised using our canonical data model but what the terms denote varies depending on which layer they are in. The layers are joined together by dynamically changing conceptual links established between the components of the layers. The KB logically groups schema objects assumed to denote the same real world concepts into clusters (*object clusters*) in the meta-data layer.

In the concept layer, terms represent *concepts* and *concept properties*. A concept can be understood as a high level knowledge abstraction which guides the classification of terms into clusters in the meta-data layer, while concept properties describe concepts. Hence a concept in the concept layer characterises a cluster in the meta-data layer. Concepts and concept properties may be linked with each other via semantic relationships such as aggregation, generalisation/specialisation, positive-association. The strength property of a link connecting a concept property and a concept represents how strongly the concept property characterises the concept, while the strength property of links connecting concepts represents how closely the concepts are related in the federation. The values of these strengths are computed by observing how schema components are related in the component DB schemas, and change dynamically as databases join and leave the federation and as views are created over the federation, to reflect how concepts are associated together with respect to the component databases and user views. The concept layer may be viewed as an ontology [Gru93] created for the federation by using the information of the federation.

The view layer stores information relevant to views generated by users over the federation. The terms in this layer represent the views as a whole (i.e. view name, owner information, date of creation, date of expiration) and their components. The nodes corresponding to terms may have attached materialisation or extent derivation rules. These materialisation rules are typically used by query processors to derive extensions of the views from the corresponding DBs. The view layer is aimed at providing a repository of reusable integrating solutions and at gathering knowledge required to determine and notify view users about DB meta-data changes that potentially affect their views.

The meta-data layer records all the schema structures generated for the component databases in the federation by organising the terms in them into object clusters. Hence in the meta-data layer, terms represent schema objects and their attributes. The classification of terms into object clusters is guided by the concepts in the concept layer and during this process concept properties in the concept layer may become new concepts. For example "publisher" may be a concept property of a concept "book" initially and may become a concept in its own right with concept properties 'name', 'address' etc. after merging a new component DB in the federation with the KB. Since our canonical data model is based on the binary-relational view of the universe [Fro82] it is flexible enough to represent a wide variety of meta-data about schema objects and provides a high degree of modifiability. The meta-data layer is intended to be used to assist users in reconciling schematic differences among semantically similar items during schema integration, where this is seen as a process undertaken by a user to establish their views of the federated database.

The DB layer maintains information about the databases whose schema objects are represented in the meta-data layer. The terms in this layer represent DBs in the federation and the properties of the terms describe the DBs. The DB layer is intended to store information necessary to access extensions of the component databases.

In the context of databases, the advantage of separation of DB elements into groups on their semantics is demonstrated in [EN84, Ken91], but no attempt is made to explain how the classification is done. Our KB is analogous to the federal dictionary in [HM85]. When a DB joins the federation its meta-data is extracted and merged with the knowledge in the KB. The initial knowledge content of the KB is typically the meta-data of the first DB joining the federation. However our system does not preclude using an appropriate ontology [Gru93] to build the initial framework of the concept layer of the KB.

4 The Workbench

Our prototype workbench is designed to support a client-server architecture. It comprises of a back-end *Knowledge Server* (*KS*) and a front-end *Graphical User Interface* (*GUI*). The back-end KS manages the KB and provides the functionality necessary for the establishment and evolution of the KB and for its role in assisting users in their integration efforts. The GUI acts as a client of the KS and invokes the services provided by the KS on behalf of the users. In addition the GUI provides services for users to access and extract meta-data from the component databases in the

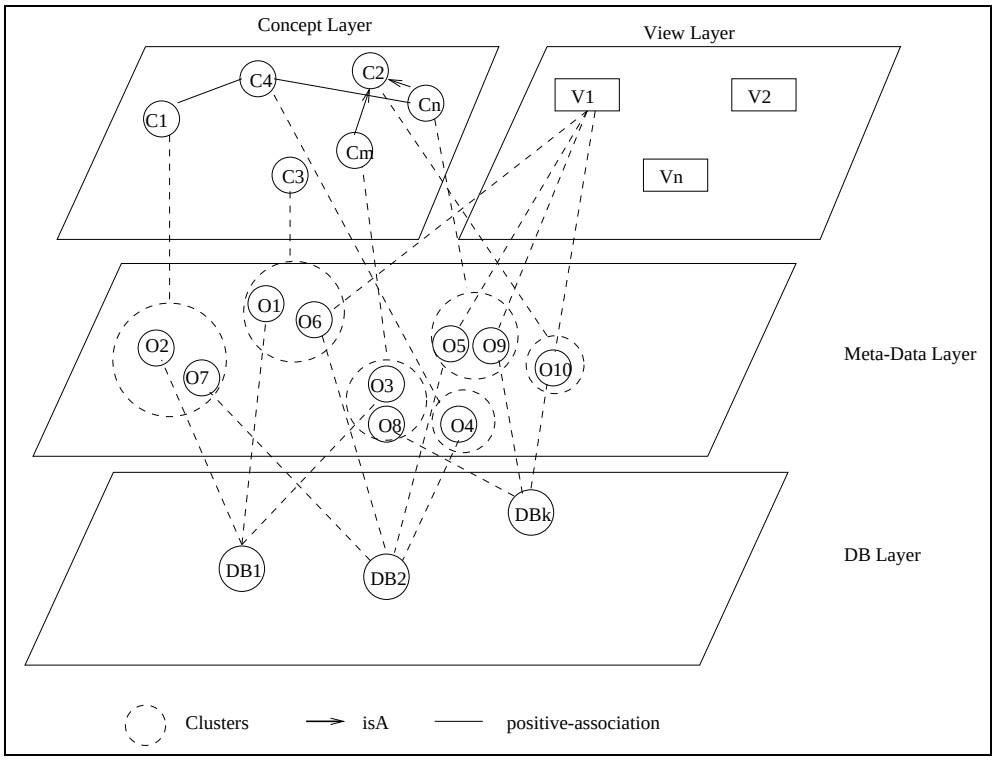

Figure 3: The Structure of the KB

federation. The services of the GUI and KS are implemented as separate plug-in tools, allowing new tools to be added and existing tools to be deleted (or modified) from the workbench with ease.

Two major tools in the GUI are the *Meta Data Extractor* (*MDE*) and the *Knowledge Browser*. The MDE accesses the individual component DBs, extracts meta-data and builds a schema structure locally for each component DB in the federation. It interacts with the component databases in the federation via JDBC drivers, thereby preserving autonomy. Users may view the schema structures and may modify and enrich them by using the knowledge browser. The knowledge browser allows users to add properties to nodes and links, change the property values, add (and delete) links and delete nodes. The users may view the schema structures as graphs by means of the knowledge browser. Since each layer of the KB is also structured by using the same data model, users can view the content of any layer of the KB or a portion of it through the same GUI. We intend to extend the functionality of the knowledge browser so that users can view the schema structures in many different conceptual models (e.g. ER, OMT), according to their preferences.

Some of the tools provided by the KS are the *Schema Analyser* (*SA*), *Concept Manager* (*CM*), *Resource Discoverer* (*RD*) and *View Manager* (*VM*). The DB owners may use the SA to analyse a schema structure to determine implicit relationships between schema elements. Our system encourages DB owners to determine such implicit relationships and to encode them explicitly in the schema structures before merging with the KB. The SA is based on clustering of similar schema elements followed by unification [Kni89] of properties of schema elements within clusters. These clustering and unification processes basically use the implementation properties of the schema elements (e.g. domain, field width, foreign key definitions) to determine their similarity. The SA builds tentative links among schema elements discovered to be similar. The strengths of the tentative links are computed by using Dice's coefficient [Rij75]. Tentative links generated by the SA can be considered as a set of propositions, that should either be proved or refuted by the DB owners. The classification of schema elements to clusters, and selecting and assigning weights (in the interval $[0,1]$) to the implementation properties of schema elements used to compute the strengths of the tentative links, are done by consulting a control file. The users may modify this file to tailor the classification and unification processes. For example, users may define a function in the control file to assign a higher weight to the schema element property "name" when the name of a schema element (e.g. title) is embedded in another (e.g. book_title). The strengths associated with the tentative links provide a measure of likelihood of such links which guides the DB owners

in making their decisions.

The CM is responsible for merging schema structures with the KB. We merge a schema structure with the KB by generating the best sub-network that spans the concepts common to the schema structure and the KB and then by unifying this sub-network with the schema structure. Hence the terms in schema structures are given interpretations not by considering them in isolation but by taking into account contexts, derived from the KB with respect to those schema terms. This process is semi-automatic and the algorithm used by the CM to merge schema structures with the KB is explained in [KGF98].

Users may use the RD to locate DBs in the federation that store similar data. The structure of the KB provides two different entry levels for the RD to start the resource discovery process: concept layer or DB layer. A user can initiate the discovery process from the concept layer by browsing and selecting a set of concepts from this layer that he thinks might be relevant for his application. In this case the RD first determines the object clusters to which the selected concepts belong. Next it navigates along the conceptual links between the concept layer and the meta-data layer to determine similar schema elements, then along the conceptual links between the meta-data layer and the DB layer to discover the corresponding DBs. Alternatively, users may start the discovery process from the DB layer by selecting a candidate DB(s) from it. In this case the RD uses the selected DB(s) to generate a set of concepts at the concept layer by moving along the conceptual links between the DB layer and the meta-data layer, then along the links between the meta-data layer and the concept layer. The generated set of concepts is used as in the former case to discover DBs in the federation that define similar schema objects.

The main function of the VM is to assist users to generate integrating views over the databases whose meta-data is available in the meta-data layer of the KB and to inform them when their views become invalid due to schema modifications. This tool provides the users with a set of operators [Duw97] that can be applied to schema definitions in the meta-data layer to create integrating views at the view layer. When the schemas of component DBs are modified, the DB owners may communicate these changes to the system by merging the new schema structure generated over the modified DB with the KB. In such cases the KS invokes the VM with the modified schema elements so that the VM checks the affected views in the view layer, invalidates them and informs their owners. The DB owners may not communicate all the modifications to the KS. Therefore any query processor using the KB also should invoke the VM with necessary information when it detects changes in the component DB schemas. The portion of the VM that checks the validity of the existing views is currently under construction.

We have implemented the KS in Quintus Prolog [Cor91] and the GUI in JAVA, and used socket connections to link them.

5 Conclusion

In this paper we have described a canonical data model and a KB to organise the knowledge characterising a loosely-coupled federation of heterogeneous databases. A workbench of tools to assist users to build this KB for a federation of databases and to generate views over the federation is also explained. Our research is guided by the following assumptions that :

- Schemas developed to represent similar real world domains typically share a set of common terms as schema object names and their attribute names.
- Semantic relationships among schema elements in a federation can be identified better by pooling meta-data of component DBs.

A KB established by using our system is more germane to the federation as it uses mainly the meta information of the DBs from the schemas and the DB owners, and more comprehensive as it stores information necessary for a wide variety of activities such as information browsing, planning/preparing for an integration, resolving schematic conflicts, accessing data, etc. A number of proposals for the creation of semantic dictionaries to help users at different stages of the integrating view creation process are reported in the literature, but we have not come across as comprehensive and extensible a KB as ours that stores information necessary for all stages of integrating view creation and maintenance. The concept layer of the KB can be considered as an ontology describing the semantic entities modelled by the component DBs in the federation. Our system incorporates a novel method of generating such ontologies for a federation from the schema definitions of the component DBs. Here it is not necessary to create an ontology for the federation and to link schema elements to the components in the ontology separately. We believe that our system will help users who build integrating views to be more productive as the KB provides, in one place, all the information necessary to generate such views, and the tools to assist in browsing, discovering, retrieving and integrating schemas known to the KB.

The architecture of the KB and the tools explained in this paper are based on a prototype system that we built to support users creating views over a set of library databases. For our initial prototype experiments

we have used three library databases each containing around 30 attributes in about 8 tables. Our experience so far has confirmed the potential value of such a KB and the re-usable knowledge it holds, in assisting users to create integrating views effectively in a loosely-coupled database federation.

Appendix

```
create table catalogued_book
  (isbn char(13),
   title char(30),
   publisher char(30),
   year number(4,0),
   classmark char(13),
   CONSTRAINTS pk_catalogued_book
   PRIMARY KEY (isbn));

create table author
  (isbn char(13)
        CONSTRAINTS fk_author
        REFERENCES catalogued_book(isbn),
   name char(30),
   CONSTRAINTS pk_authors
   PRIMARY KEY (isbn));

create table loan_status
  (status char(1),
   loan_period number(3,0),
   max_num_books number(2,0),
   CONSTRAINTS pk_loan_status
   PRIMARY KEY (status));

create table borrower
  (id number(10,0),
   name char(30),
   address char(30),
   status char(1)
       CONSTRAINTS fk_borrower
       REFERENCES loan_status(status),
   CONSTRAINTS pk_borrower
   PRIMARY KEY (id));

create table loan
  (book_no char(10),
  isbn char(13)
      CONSTRAINTS fk_book
      REFERENCES catalogued_book(isbn),
  shelf char(10),
  borrower_id number(10,0)
      CONSTRAINTS fk_book2
      REFERENCES borrower(id),
  date_due_back date,
  CONSTRAINTS pk_book
  PRIMARY KEY (book_no));

create table reservation
  (isbn char(13)
     CONSTRAINTS fk_reservation1
     REFERENCES catalogued_book(isbn),
   id  number(10,0)
     CONSTRAINTS fk_reservation2
     REFERENCES borrower(id),
   date_reserved date);
```

The set of SQL commands used to create a sample ORACLE database

References

[BHP92] M. W. Bright, A. R. Hurson, and S. H. Pakzad. A taxonomy and current issues in multidatabase systems. *IEEE Computer*, 25(3), March 1992.

[CA97] S. Castano and V. De Antonellis. Semantic dictionary design for database interoperability. In *Proceedings of the 13th International Conference on Data Engineering*, pages 43–54, Birmingham, U.K., April 1997.

[CHS91] C. Collet, M. N. Huhns, and W. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, pages 55–62, December 1991.

[Cor91] Quintus Corporation. *Quintus Prolog, Release 3, Language and Library.* Quintus Corporation, 1991.

[CS91] A. Chatterjee and A. Segev. Data manipulation in heterogeneous databases. *SIGMOD RECORD*, 20(4), December 1991.

[DKW+91] S. Dao, D. M. Keirsey, R. Williamson, S. Goldman, and C. P. Dolan. Smart data dictionary: A knowledge-object-oriented approach for interoperability of heterogeneous information management systems. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pages 88–91, Kyoto, Japan, April 1991.

[Duw97] R. M. Duwairi. *Views for Interoperability in a Heterogeneous Object-Oriented Multidatabase System.* PhD thesis, Department of Computer Science, University of Wales College of Cardiff, April 1997.

[EM97] B. EagleStone and N. Masood. Schema interpretation: An aid to schema analysis in federated database design. In *Engineering Federated Database Systems (EFDBS97), Proceedings of the International CAiSE97 Workshop*, University of Magdeberg, Germany, June 1997.

[EN84] R. Elmasri and S. Navathe. Object integration in logical database design. In *Proceedings of the 1st International Conference on Data Engineering*, pages 426–433, Los Angeles, April 1984.

[Fro82] R. A. Frost. Binary-relational storage structures. *The Computer Journal*, 25(3):358–367, 1982.

[GMI97] A. Goni, E. Mena, and I. Illarramendi. Querying heterogeneous and distributed data repositories using ontologies. In *Proceedings of the 7th European-Japanese Conference on Information Modelling and Knowledge Bases (IMKB'97)*, Toulouse, France, May 1997.

[Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[HM81] D. Heimbigner and D. McLeod. Federated information bases - a preliminary report. In *Infotech State of the Art Report, Databases*, pages 223–232. Pergamon Infotech Limited, Maidenhead, Berkshire, England, 1981.

[HM85] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.

[HM93] J. Hammer and D. McLeod. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):51–83, March 1993.

[Ken91] W. Kent. Solving domain mismatch and schema mismatch problems with an object-oriented database programming language. In *Proceedings of the 17th VLDB Conference*, pages 147–160, Barcelona, Spain, September 1991.

[KGF98] D. D. Karunaratna, W. A. Gray, and N. J. Fiddian. Establishing a knowledge base to assist integration of heterogeneous databases. In *Proceedings of the 16th British National Conference on Databases(BNCOD16)*, Cardiff, U.K., July 1998. Springer-Verlag. To be published.

[Kni89] K. Knight. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1), March 1989.

[KS91] W. Kim and J. Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, pages 12–18, December 1991.

[KS93] V. Kashyap and A. Sheth. Schema correspondences between objects with semantic proximity. Technical Report DCS-TR-301, Department of Computer Science, Rutgers University, October 1993.

[Lit88] W. Litwin. From database systems to multidatabase systems: Why and how. In *Proceedings of the 6th British National Conference on Databases(BNCOD6)*, pages 161–188, Cardiff, U.K., 1988.

[LMR90] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[MBP95] S. Milliner, A. Bouguettaya, and M. Papazoglou. A scalable architecture for autonomous heterogeneous database interactions. In *Proceedings of the 21st VLDB Conference*, pages 515–526, Zurich, Switzerland, 1995.

[McC82] J. L. McCarthy. Metadata management for large statistical databases. In *Proceedings of the 8th VLDB Conference*, pages 234–243, Mexico City, Mexico, September 1982.

[Mil95] G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11), November 1995.

[MKSI96] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems(CoopIS'96)*, Brussels, Belgium, June 1996.

[PBE96] E. Pitoura, O. Bukhres, and A. K. Elmagarmid. Object-oriented multidatabase systems: An overview. In O. Bukhres and A. K. Elmagarmid, editors, *Object-Oriented Multidatabase Systems, A Solution for Advanced Applications*. Prentice Hall, 1996.

[Rij75] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth and Co(Publishers) Ltd, 1975.

[RS94] A. Rosenthal and L. J. Seligman. Data integration in the large: The challenge of reuse. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, September 1994.

[SGN93] A. P. Sheth, S. K. Gala, and S. B. Navathe. On automatic reasoning for schema integration. *International Journal of Intelligent and Co-operative Information Systems*, 2(1):23–50, March 1993.

[She91] A. P. Sheth. Semantic issues in multidatabase systems. *SIGMOD RECORD*, 20(4):5–9, December 1991.

[SK92] A. Sheth and V. Kashyap. So far (Schematically) yet so near (Semantically). In *Proceedings of TC2-IFIP WG 2.6 Conference on Semantics of Interoperable Database Systems (DS-5)*, pages 283–312, Lorne, Victoria, Australia, November 1992.

[SL90] A. P. Sheth and J. P. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SM91] M. Siegel and S. E. Madnick. A metadata approach to resolving semantic conflicts. In *Proceedings of the 17th VLDB Conference*, Barcelona, Spain, September 1991.

[SPD92] S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1:81–126, July 1992.

[SSU90] A. Silberschatz, M. Stonebraker, and J. D. Ullman. Database systems: Achievements and opportunities. *SIGMOD RECORD*, 19(4):23–31, December 1990.

[YSDK91] C. Yu, W. Sun, S. Dao, and D. Keirsey. Determining relationships among attributes for interoperability of multidatabase systems. In *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, pages 251–257, Kyoto, Japan, April 1991.