Use of an Object-based system with reasoning capabilities to integrate relational databases

Goñi A., Illarramendi A., Blanco J.M., Mena E.

Facultad de Informática, Universidad del País_Vasco. Apdo. 649,
20.080 San Sebastián Basque Country
e-mail: jipileca@si.ehu.es
phone: + 34 43 218000

Abstract

The integration of heterogeneous and autonomous information sources is a requirement for the new type of cooperative information systems. In this paper we show the advantages of using a terminological system for integrating pre-existing relational databases. From the resulting integrated schema point of view, using a terminological system allows for the definition of semantically richer integrated schema. From the integrated schema generation process point of view, the use of a terminological system permits the definition of a more consistent, broad and automatic process. Last, from the query processing point of view, terminological systems provide interesting features for incorporating semantic and caching query optimization techniques. The advantages are presented in detail for each main step of the integration process: translation, integration and query processing.

1 Introduction

Within most organizations there exists many heterogeneous databases, that have been defined independently, which store data that are somehow related. A global and uniform treatment of all these data, maintaining at the same time the autonomy of the component systems, will bring the organizations the opportunity of improving the management of their information.

In the literature several attempts to solve the interoperability problem and in particular the database interoperability can be found. However, a standard solution that gives an answer to all requirements has not been proposed yet. In [SPD92] an overview of different proposals is presented.

In this paper we wish to present our experience of building a system that allows the integration of heterogeneous and autonomous relational databases using a terminological system. This last type of systems, more recently termed description logics (DL), allow for the specification of classes using intensional descriptions phrased in terms of the necessary and sufficient properties that must be satisfied by their instances. The key difference between such descriptions and the earlier class specifications is that such a definition can be used to recognize instances of the class, or to infer answers to queries in situation where there is incomplete knowledge [Bor92]. Indeed the main focus of the paper is on presenting the advantages of using a terminological system for the main steps of the integration process: a *translation* step for obtaining a uniform and richer representation of the schemata that must be integrated; an *integration* step for creating an integrated schema and a *query processing* step for giving answers to the queries formulated over the integrated schema.

Among the related work that also use a terminological system for the system integration process we can mention [BS92], [ACHK93], [BST$^+$93] and [SGN93]. In [BS92] the use of taxonomic reasoning techniques to support the conceptual design of schemata is proposed. With this aim they have incorporated taxonomic reasoning techniques to some well-known semantic data models. They maintain that the designed schemata will be more consistent. In [ACHK93] an existing terminological system (LOOM) is used as a model to describe database schemata and, a system that uses LOOM to provide efficient access to a relational database is described. However, the schemata integration task is somehow limited; they do not allow relations between data elements that belong to different schemata. Our proposal is richer in this sense. In [BST$^+$93] it is argued that the exploitation of pre-existing databases using a terminological system (CLASSIC) allows the generation of new information sources. Nevertheless, they assume that the integrated schema is defined previously and so the integration task is reduced to the mapping process among the local and the integrated schemata. Finally, in [SGN93] CANDIDE, a DBMS based on description logics, is used as a tool to automate a significant part of the schemata integration process.

The main advantages that the use of a terminological system provides for the integration process and that are explained in this paper are the following: the possibility of defining semantically rich integrated schema; a wider range of translation and integration types; the automatic verification and derivation of new relationships among classes; the automatic detection of inconsistent queries and also reformulation of them; and last, the definition of cached concepts and automatic discovering of cached queries.

In the remainder of this paper we present first the general framework for the system components explained in this paper followed by a brief introduction to terminological systems. Finally we explain the advantages of using a terminological system for the translation, integration and query processing.

2 Work's framework

The general framework of the designed federated system that allows the integration of heterogeneous relational databases using a terminological system is presented in this section. In the framework three main components can be distinguished[1]: Translator, Integrator and Query Processor.

The Translator component produces a terminology from a conceptual schema (or a subset of it, called exported schema) of a component database. The resultant terminology will be semantically richer than the source schema, therefore this component has to capture, with the PRI's help, semantics that are not expressed explicitly.

The Integrator component produces a federated terminology by integrating a set of terminologies previously obtained by the Translator component. During the integration process a set of correspondences between data elements of the terminologies that must be integrated will be defined by the PRI and new ones can also be deduced by the system.

[1]Actually, there is another component, called Controller, that is not discussed in this paper, because it does not take advantage of any particular feature of terminological systems. The Controller responds automatically, i.e., without user intervention, to design changes made in the schema of a component database that affect the federated terminology [BIPG92].

The Query Processor component obtains the answer to the user formulated queries over the federated terminology by accessing the databases. This component has two kind of modules: the Global Query Processor and the Local Query Processor. The first one optimizes knowledge base queries, finds out which databases contain the requested information, decomposes a query into subqueries that will run over different databases, and reconstructs the answer from the different results obtained for the subqueries. The main advantages that the use of a terminological system provides for this module are: a semantic optimization of queries using the classification mechanism[2]; support for defining and identifying cached data; and the possibility of giving intensional answers. Moreover, the goals of the second module are to make local optimizations and to find the answer for the subqueries.

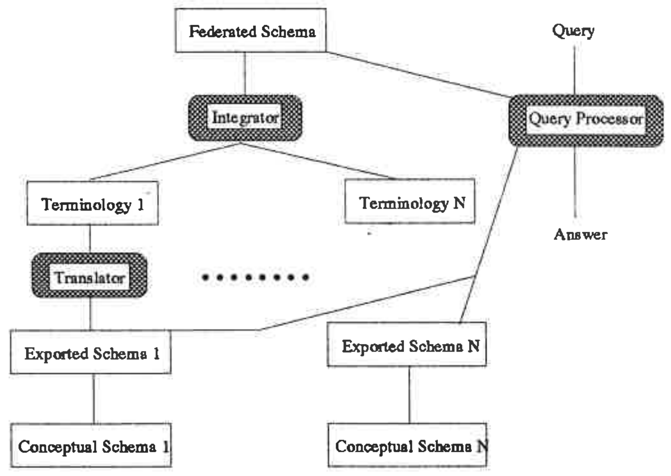


Figure 1: Work's framework

3 A brief introduction to terminological systems

In general, two different components can be distinguished in this type of systems: the *terminological* component, which is used to describe the knowledge

[2]The meaning of the classification mechanism for the terminological systems is explained in the next section

(the terminology) and the *assertional* component, which is used to create instance objects that represent the beliefs of the system.

In the terminology, two main types of data elements are included: concepts and roles[3]. A *concept* groups individual elements of the real world. For example, the concept *employee* represents the set of employees of a company. However what distinguishes this notion of concept from the class specification in semantic data models or object-oriented databases, is that it is possible to describe concepts using intensional descriptions phrased not only in terms of necessary properties that must be satisfied by their instances (in this case the concept is called a *primitive concept*) but also in terms of necessary and sufficient properties (in this case the concept is called a *defined concept*) [Bor92]. This is accomplished using a language for describing primitive concepts and then combining them into composite ones.

Examples: The concepts *client* and *manager* can be described as primitive concept in the following way:
client :<[4] *anything*[5].
manager :< *employee and Category:'BS degree'*

This means that a *manager* is an employee which has a BS degree, but there can exist employees with a BS degree that are not managers.

slowpayer can be described as a defined concept:
slowpayer :=[6] *client and all(Payment,ge(40))*

slowpayer are *clients* who make payments in a period greater or equal to 40 days, and because it is a defined concept, any client who makes payments in a period greater or equal to 40 is a *slowpayer*.

Roles represent binary relationships between concept instances and other instances or values. A role can be seen as a predicate with two arguments: the *domain* (the concept with which is associated) and the *range* (the type of values that it can have).

Example: The roles *Payment* and *Buys* can be described as
Payment :< *domain(client) and range(number)*,
Buys :< *domain(client) and range(product)*.

Both roles are associated with the concept *client*; *Payment* takes numeric values and *Buys* has as values instances of the concept *product*.

[3]We use the BACK system notation [PSKQ89]. In other systems the components are denoted in a different way, for example as classes and attributes in [BBMR89].

[4]:< is used, according to the BACK syntax, to describe primitive concepts.

[5]*anything* is a concept such that any instance can belong to it.

[6]:= is used, according to the BACK syntax, to describe defined concepts.

Two important features in terminological systems are the notions of *subsumption* and *classification*. One concept *subsumes* another one if in all possible circumstances, any instance of the second one must be in the first one. In a terminological system, it is possible to know whether one concept *is subsumed* by another one simply by looking at the definition of the concepts, without accessing to the instances. The *classification* mechanism consists on discovering the subsumption relationships between concepts when a new concept is declared, i.e. the new concept is automatically located into the hierarchy of terms, therefore concepts viewed as composite descriptions, can be reasoned with and are the source of inferences.

Example: Introducing a new defined concept *government* described as *government := client and all(Payment,ge(60))* in the knowledge base that contains the concepts *client* and *slowpayer* presented above, the system will automatically classify it under the *slowpayer* concept.

When using a terminological system to build a federated database system, the resultant federated schema is represented by a terminology. But the actual data are stored in the underlying databases. For this reason it is necessary to define a linking information between the federated terminology and the local databases.

Finally, making a brief comparison between the well-known E/R model and the terminological systems [WS92] we could find a resemblance among the notions of entity and attribute of the E/R model with the notions of concept and role respectively of the terminological systems, but, in the latter case, concepts and roles are attached with intensional descriptions. In the terminological systems a specific constructor does not exist, as it exists in the E/R model, to represent relationships. These are represented through the roles. Moreover, terminological systems provide structuring mechanisms such as specialization and the previously mentioned classification mechanism that are not supported by the basic E/R model. Lastly, terminological systems support an internal identification of the instances which is different to the E/R, which provides an external identification using the notion of key.

4 Translation Process

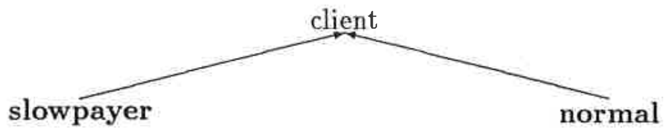
The first requirement to do an integration of heterogeneous databases is to define them in a uniform way, that is using the same data model, called a canonical model. It is assumed that this canonical model should have an expressive power higher than that offered by the relational model in order to support an interesting integration process, hence most of the works in the area use semantic or object-oriented data models. In our case we use a terminological system. The main advantages that its use provides in the translation process can be summarized in two aspects: the definition of semantically richer schemata and the definition of complex translation types.

Although these advantages can also be obtained somehow using semantic or object-oriented data models, our goal is to show that they are particularly relevant for dealing with a terminological system.

Definition of semantically richer schemata. This can be achieved using the structural mechanisms provided by a terminological system such as generalization/specialization that permit the definition of conceptual hierarchies. For example, the relation *client* defined in a relational schema as

client(c#,name,address,payment)

that contains data about all the clients of an enterprise could be translated into the following hierarchy of concepts (see figure below). The concepts *slowpayer* and *normal* are specializations of the concept *client.*



The semantics expressed explicitly in the hierarchy is higher than that represented by the flat relation. In order to be able to obtain these hierarchies, properties about dependencies: inclusion, exclusion and functional; information about null values or information about domain values of attributes can be used. For example, in the relation *client* the domain values that *clients* qualified as *slowpayer* take for the attribute *payment* must be greater than 30, and for the *normal clients* less than 30. In this example, information about domain values of the attribute *payment* has been used. In [BIGP94] we present in detail how the kinds of knowledge mentioned before can be used in order to obtain rich hierarchies.

Furthermore, the use of a terminological system facilitates the process of building these hierarchies because, due to the classification provided by them, the *is-a* relationships are handled automatically. This means that the PRI does not need to know the exact place in the hierarchy where a new concept should be introduced. The terminological system will automatically discover the correct place.

Last, the feature provided by terminological systems of attaching descriptions to concepts in terms of necessary or necessary and sufficient properties enrich also from a semantic point of view the resulting schema. For example, the concepts *slowpayer* and *normal* can be described as

```
slowpayer := client and all(payment, ge 30)
normal := client and all(payment, le 30)
```

Definition of complex translation types. This advantage is also related to the fact that concepts are associated with intensional descriptions. Intensional descriptions allows one to define translations in terms of semantic properties that are not supported by other approaches. For example, the concept *active-employee* of a hierarchy could be defined as

active-employee := *employee and atleast(2,participates)*

where *employee* makes reference to a relational table

and *atleast(2,participates)* expresses a property that must be verified by the tuples in order to be considered them as *active-employees.* Complex objects can be defined over concepts and roles in the hierarchy. Notice that this issue allows a wide range of translation types.

5 The Integration process

Once the terminologies have been obtained from the component databases, the next step consists in integrating the different terminologies in order to obtain an integrated schema.

In general, an integration process requires first of all, to define correspondences among data elements (concepts and roles) of the terminologies that must be integrated and then the application of some integration rules. In the literature several approaches that deal with the system integration process have been presented [BEM92]. In the following we show the advantages that the use of a terminological system provides for this process. These advantages can be summarized in the following points:

- the possibility of automatically discovering errors in the definition of correspondences between data elements expressed by the PRI;
- the automatic inference of new correspondences between data elements not explicitly defined by the PRI; and

- the possibility of defining correspondences between data elements using intensional descriptions.

In the following subsections we describe these advantages in more detail.

5.1 Verification of correspondences

As mentioned before, in order to carry out an integration process, first of all, correspondences among data elements of different terminologies must be defined. It is widely accepted that it is almost impossible to obtain them in an automatic way, because they are related with the semantics of the Universe of Discourse where the databases have been designed. Therefore it is necessary to have the collaboration of the PRI. Some work e.g [LNE89], require the PRI to define correspondences among roles, while others, e.g. [SPD92], require the definition of correspondences among concepts. Nevertheless, the majority agree on using the Real World State (RWS) notion, i.e., the semantics of correspondence assertions among data elements of different schemata is defined referring to the real counterpart of elements in the schemata. The four types of commonly used correspondences are: *equivalence, inclusion, overlapping* and *disjoint*.

In the case of dealing with a terminological system, correspondences defined in terms of a RWS notion can be verified using the intensional descriptions associated with the concepts among which the correspondence is established. For example, if the following two concepts are defined in two different terminologies.

```
good := client and atleast(1,payment) and all(payment,le 30)
Government := client and atleast(1,payment) and all(payment,ge 60)
```

and the PRI expresses that *good equivalent Government* because he or she misunderstood the semantics associated with the concepts, the terminological system will inform him or her that it is not correct because the descriptions are incompatible.

5.2 Automatic derivation

One important feature provided by terminological systems is the classification mechanism. This mechanism can play an important role in the integration process deriving new relationships between data elements not defined explicitly by the PRI. For example, during the integration process of two terminologies (see figure 2), if the PRI asserts that *t1_client equivalent t2_client*[7] as a result of applying the corresponding integration rule only one concept *client* will appear in the resulting integrated schema.

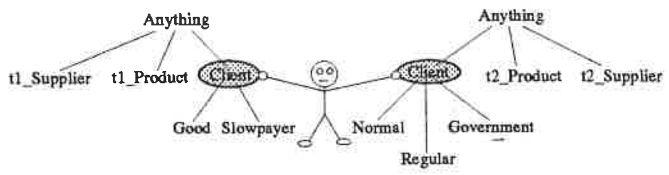


Figure 2: Asserting an equivalence

Moreover, after the redefinition process, the terminological system will automatically discover (see figure 3) that *good* and *normal* are equivalent concepts, and that *slowpayer* subsumes *regular* and *Government* (the descriptions of *normal regular* and *Government* are shown in figure 4).

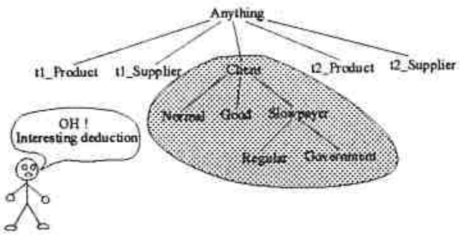


Figure 3: Resulting integrated schema

5.3 Correspondences based on descriptions

In the majority of work in the integration area the correspondences are defined using the notion of RWS, that is based on the extensions of the data elements that are being related. However, using a terminological system, more complex correspondences can be expressed using semantic properties.

[7]This correspondence is correct in terms of the descriptions associated with the concepts *client*.

```
client :< anything
t2_product :< anything
t2_supplier :< anything
payment :< domain(client) and range(number)
buys :< domain(client) and range(t2_product)
normal := client and atleast(1,payment) and all(payment,le 30)
regular := client and atleast(1,payment) and all(payment,ge 40) and all(payment,le 60)
Government := client and atleast(1,payment) and all(payment,ge 60)
```

Figure 4: BACK descriptions of one of the terminologies (*t2*)

This is possible due to the facility provided of adding intensional descriptions to the concepts definitions. For example, the PRI could define the correspondence *engineer* $\equiv$ *employee and all(salary,ge 5.000.000))* to express that *engineers* of one terminology are employees (according to their definition in other terminology) with a salary greater than 5.000.000. It is not possible to define this type of correspondences by using directly the RWS notion utilized in other work. This the specification of concepts using intensional descriptions permits a richer integration process.

Finally, notice that the full power of the previous advantages for translation and integration is not supported by most semantic or object-oriented data models. However, in the case of dealing with an object-oriented database models it is possible to define an operational integration, that is, among operations at different levels, while terminological systems do not provide this possibility.

6 Query Processing

Once an integrated schema has been obtained, it provides to the users with an integrated and global view of the data stored in pre-existing databases that will be used by them to formulate queries over. To find the answers to these queries in an efficient way is the goal of the query processing step.

Although there has been a lot of research into the problem of translation and integration of schemata in order to obtain an integrated schema, the problem of query processing against integrated schema has not be investigated so much. Furthermore, query techniques developed in the area of distributed database cannot been applied directly to the interoperability context, as several authors claim ([OV92, Day85, LOG92]) because of the heterogeneity and autonomy of the component databases.

The main advantages that the use of a terminological system provides for the Global Query processor (notice that Local Query processors are developed for the local database systems) are summarized in the following points:

- a semantic optimization of queries using the classification mechanism;
- support for defining and identifying cached data; and
- the possibility of given intensional answers.

6.1 Use of the classification mechanism to semantically optimize the queries

In the database area, semantic query optimization methods exploit domains knowledge such as that expressed by integrity constraints, hierarchies, etc. to detect inconsistent queries or to transform a user formulated query into another one with the same answer, that is semantically equivalent, but that can be processed more efficiently. These semantic optimization methods are external to the database systems and are defined as a special purpose mechanism. Using a terminological system, it is possible to do semantic query optimization using the reasoning capabilities of these systems. The classification mechanism of a terminological system allows the detection of inconsistent queries and the reformulation of them.

Concerning the first issue, consider the following example. If the query[8]

[rf(buys)] for getall Government and payment: 50

asking for the products bought by clients qualified as Government that have paid in the 50th day is formulated, the terminological system will detect it as inconsistent. The reason for that is that the concept *Government* was defined (see figure 4) as client with all payments greater or equal to 60 and hence the query contradicts this definition. It is very important to detect inconsistent queries in an interoperable context because costly processes of accessing the underlying databases need to be avoided.

Concerning the second issue, reformulating queries, this can be achieved by finding the *most specific superconcepts* related to the queries. For example, suppose that the following query is formulated:

getall client and all(payment, ge 60)

then, the most specific superconcept will be *Government* (according to the terminology described in figure 4) and the query can be expressed as:

[8]Here we use the BACK query language notation.

getall Government

A parent concept C_i, that does not appear in the query definition, is the most specific superconcept that could have been expressed for the query definition. The user does not need to remember all the concepts defined in the terminology to formulate more efficient queries but the Generator Processor can find them. Nevertheless, as occurs in the database context, notice that these reformulations must be analyzed later on to discover if they are interesting in terms of reducing the access time, because it is not always true that accessing the tuples represented by the concept *Government* is more efficient than accessing those represented by *client.*

6.2 Maintaining of cached concepts

In a client-server architecture it is worth having cached data in the client nodes to avoid accessing the server nodes every time a query is made. In [DR93] they show the advantages of having an Enhanced Client-Server architecture where they incorporate relational DBMSs in the client nodes to maintain cached data. In our case we can use the same terminological system to store cached data. The idea is to download part of the data in the underlying databases located in the server nodes into a terminological cache in the client node. Moreover, when there exist space problems, concepts can be cached with incomplete information.

After a process of identifying the data that are worth caching, they can be grouped under a concept and later introduced into a terminology by using the capability of defining rules provided by terminological systems[9]. For example, suppose that we are interested in catching *engineer* described as

engineer := employee and all(salary,gt(5.000.000))

it could be declared as cached defining the rule

engineer => cached

It is obvious that this would not be useful if one could not ask if a concept is cached or not. But the subsumption mechanism of the terminological system provides this capability. In fact, if the concept *cached* subsumes the one about which we are asking, then that concept is cached. For example, after the declaration of the previous rule, if the following query

getall employee and all(salary,gt(10.000.000))

[9]With the use of the rules, non-definitional information can be added to the terminologies.

is formulated, then the system will recognize it[10] as cached, because

subsumes(cached,employee and all(salary,gt(10.000.000)).

Last, notice that the user does not need to know the existence of the generated cached concepts to formulate his or her queries because the Global Query processor will detect the relationship between these concepts and the user queries.

6.3 Intensional answers

User queries are usually answered by giving the set ~~of~~ instances that satisfy the conditions in the query. They are considered as extensional answers. However, when working with terminological systems it is also possible to give answers in terms of descriptions that satisfy the instances. In this case they are considered as intensional answers. This can be done in two ways:

1. By giving the most specific formulation. For example for the query

   getall employee and atleast(1,participates)

   the answer would be *active-employee* telling the user that the employees he or she is asking for are those qualified as *active-employee.* Of course, in the next step he or she could ask for the concrete instances.

2. By giving the extended query formulation. This is possible by using the concept definitions instead of concept names. For example for the query

   getall Government and all(payment,ge 75)

   the extended query formulation would be

   anything and client and all(payment,ge 60) and all(payment,ge 75).

   This extended query formulation could be interesting for example for inconsistent queries in order to know the reason for this.

7 Conclusions

In the area of information systems the problem of integrating different application environments has attracted substantial attention. Of special interest is the integration of database systems with knowledge base systems in order to take advantage of the additional features provided by latter, without

[10]Remember that queries are treated as concepts in terminological system.

giving up the services provided by databases. The practical benefits of this type of integration apply to many different contexts, such as access to pre-existing databases when working with knowledge-based applications, allowing databases to provide persistent object support for knowledge bases, and the coexistence of different autonomous databases under a global integrated schema by using one type of knowledge based system. In this paper we have concentrated in this last point. We have shown the advantages of using a terminological system for the integration of pre-existing relational databases.

In summary we can say that from the integration point of view, terminological systems allow for consistent integration process; and from the query processing point of view they provide mechanisms to support query optimization techniques such as semantic query optimization and or caching optimization techniques.

Acknowledgements

This work was partially supported by the Commission of the European Communities and was part of the Esprit project 5210 AIMS which involved the following participants: Datamont(I), Non Standard Logic(F), Technische Universität Berlin(D), Deuteschen Herzzentrum Berlin(D), ONERA-CERT(F), Quinary(I) and Universidad del País Vasco(E).

References

[ACHK93] Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[BBMR89] A. Borgida, R.J. Brachman, D.L. McGuinness, and L.A. Resnick. Classic: A structural data model for objects. In *Proceedings ACM SIGMOD-89, Portland, Oregon*, 1989.

[BEM92] O.A. Bukhres, A.K. Elmagarmid, and J.G. Mullen. Object-oriented multidatabases: Systems and research overview. In *Proc. of the Int. Conf. Information and Knowledge Management CIKM-92 Baltimore USA*, 1992.

[BIGP94] J. M. Blanco, A. Illarramendi, A. Goñi, and J. M. Pérez. Using a terminological system to integrate relational databases. In *Information Systems Design and Hypermedia.* Cepadues-Editions, 1994.

[BIPG92] J. M. Blanco, A. Illarramendi, J. M. Pérez, and A. Goñi. Making a federated database system active. In *Database and Expert Systems Applications.* Springer-Verlag, 1992.

[Bor92] A. Borgida. From type systems to knowledge representations: Natural semantics specifications for description logics. *Intelligent and Co-operative Information Systems*, 1(1), 1992.

[BS92] S. Bergamaschi and C. Sartori. On Taxonomic Reasoning in Conceptual Design. *ACM Transactions on Database Systems*, 17(3):385–421, 1992.

[BST$^+$93] R.J. Brachman, P.G.. Selfridge, L.G. Terveen, B. Altman, A. Borgida, F. Halpner, T. Kirk, A. Lazar, D.L. McGuinnes, and L.A. Resnick. Integrated support for data archaelogy. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):159–185, 1993.

[Day85] U. Dayal. *Query Processing in a Multidatabase System*, pages 81–108. Springer-Verlag, 1985.

[DR93] A. Delis and N. Roussopoulos. Performance comparison of three modern DBMS architectures. *IEEE Transactions on Software Engineering*, 19(2), February 1993.

[LNE89] J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE TOSE*, SE-15(4), April 1989.

[LOG92] H. Lu, B. Ooi, and C. Goh. On global multidatabase query optimization. *SIGMOD RECORD*, 21(4), December 1992.

[OV92] M. T. Ozsu and P. Valduriez. *Distributed Databases: Principles and Systems.* Prentice Hall, 1992.

[PSKQ89] C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. The BACK system revisited. Technical University Berlin, September 1989.

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

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

[WS92] W.A. Woods and J.G. Schmolze. The KL-ONE family. *Computers Math. Applic.*, 23(2):133–177, 1992.