1989 WORKSHOP ON HETEROGENOUS DATABASES

December 11-13, 1989 — Northwestern University, Evanston, IL

Sponsored by NSF
In cooperation with Northwestern University and
IEEE-CS Technical Committee on Distributed Processing

Monday, December 11, 1989

8:45 - 9:15	Workshop Registration
9.15 - 10:45	**Opening Session** "Research Directions for Integrating Heterogenous Databases." W. Kim, MCC, (invited talk) "A Model for Computer Life." Witold Litwin, INRIA/Stanford University, (invited talk)
10:45 - 11:15	Coffee Break
11:15 - 12:45	**Session 2: Query Processing** (Chair: Arnon Rosenthal) *Federated Approximations for Heterogenous Databases.* O.P.Buneman, S.B.Davidson and A. Waters, University of Pennsylvania *Resolving Problems of Semantic Heterogeneity in Relational Database Systems.* L.G. DeMichiel, Hewlett-Packard Laboratories *An Intelligent Interface for Query Specification to Heterogeneous Databases.* D.J. Weishar and L. Kerschberg, George Mason University
12:45 - 14:15	Lunch - (Allen Center)
14:15 - 15:45	**Session 3: Autonomy** (Chair:Ahmed Elmagarmid) *The Distributed Operation Language for Multi-System Applications.* M. Rusinkiewicz and A. Elmagarmid, Purdue University *Heterogeneous Database Systems: Muse/Levels of integration.* B. Holtkamp, D. Hsiao and V. Lum, Naval Postgraduate School *CISL: Composing Answers from Disparate Information Systems.* S.E. Madnick, Y.R. Wang, et.al., Massachusetts Institute of Technology
15:45 - 16:15	Coffee Break
16:15 - 17:45	**Session 4 : The Object-Oriented Approach** (Chair: Frank Manola) *An Object-Oriented Approach to the Interconnection of Heterogeneous Databases.* E. Bertino, C. N. R. Pisa, M. Negri, Universita' di Brescia, G. Pelagatti and L. Sbattella, Politecnico di Milano *Control Mechanisms in an Object Management System for Supporting Interoperability of Heterogeneous Components.* S.Heiler , Xerox Advanced Information Technology *Object-Oriented to Relational Inter-Schema Meta-Translation.* A. Ramfos, N.J. Fiddian and W.A. Gray, University of Wales, U.K.

C. 26

An Object-Oriented Approach to the Interconnection of Heterogeneous Databases

E.Bertino (*), M.Negri(#), G.Pelagatti(#+), L.Sbattella(+)

(*) Istituto di Elaborazione dell'Informazione - C.N.R. Pisa (Italy)
(#) Dipartimento di Automazione Industriale - Universita' di Brescia (Italy)
(+) Dipartimento di Elettronica - Politecnico di Milano (Italy)

Extended Abstract

1. Introduction

As an organization grows, the need for combining information stored in pre-existing databases increases. Furthermore, the amount of information which is available in public databases is increasing. Therefore there is a need for suitable tools that support coordinated access to data stored in distributed, heterogeneous, autonomous *data repositories*. We use the term data repository to emphasize the fact are not necessarily managed by database management systems, but by a broad spectrum of systems, for example information retrieval systems [Salt 83], multimedia systems [Bert 88a], file systems, mail systems. In general, the situation is that all data relevant to an organization are managed by different systems that are not well integrated.

Several reasons for this are discussed in [Conn 88]. The first is that organizations evolve over time. This influences the way the data are managed within the organizations. In fact, the choice of a data management system depends on the application requirements and on the available technology. As these evolve over time, an organization ends up having several, possibly heterogeneous, data management systems. Second, in certain cases the choice of different data management systems is dictated by performance reasons. To provide adequate performance it may be necessary to maintain the data in different data repositories with different capabilities, structures, and organizations. Finally, not all the data sources belong to the organization using them. This is the case, for example, of public data banks. In this situation, the organization does not have any control on how the data are organized and presented to the users. Therefore, applications needing data from several data repositories have to bridge the gap among the various systems. This may be rather complicated due to differences in data models, or to the lack of data models in some systems, query languages, operational capabilities, such as authorization, recovery, and concurrency control.

Previous research on systems to support multiple data repository access can be classified into two categories. The first category concerns work on systems that support integrated access, creation, and maintenance of distributed homogeneous database systems, such as the systems described in [Will 81] and [Ston 77]. The second category concerns work on the integration of heterogeneous databases. The heterogeneity can be at several levels:

- Logical: this means that the various databases have been designed independently; that is, the same entity may have been modeled in different ways in the various databases. Therefore two databases may be logically heterogeneous even if the DBMSs managing them are homogeneous. Relevant work in this area includes work on database integration [Bati 86], [Motr 87]. In general, the database integration problem is facilitated if the data model used to describe the integrated schema is a semantically rich model.

- Data management system: this means that the systems managing the data have different data models and/or operational capabilities.

- Operating system and hardware.

We consider only the first two levels to be in the scope of heterogeneous database systems. Heterogeneity at the operating system and machine level are (or should be) handled by the communication subsystem. Therefore, in the remainder of this discussion we will not consider the third level.

Another classification is *integrated* vs *federated* systems. In general an integrated system has a *global integrated schema* describing all the data stored in the local databases. Furthermore, it is possible to execute from one site all operations on data stored at any other site. A federated system is instead one where each local database is independently created and administrated. Usually, there are limitations about the distributed operations that can be executed. For example, updates may only be executed locally. Furthermore, each system may make available to remote applications, or *export*, only portions of the local data. A federated system, as discussed in [Litw 86], may or may not have a global integrated schema. For example, MULTIBASE [Land 82] provides a global integrated schema which is defined by using a view mechanism [Daya 84]. This view mechanism allows the resolution of different types of semantic conflicts that arise when integrating different local schema. Other federated systems, such as MRDSM [Litw 85], do not provide any global schema at all. These systems are sometimes referenced to as *multidatabase* systems.

The user accessing a multidatabase system is therefore aware that he uses multiple databases. What a multidatabase system offers is the possibility of interfacing different databases by using a single data access and manipulation language (called *global* or *common DML*). However, a multidatabase system does not solve semantic inconsistencies among the various databases. These must be solved at application levels. We not that a system can be federated even if all the participating DBMSs are homogeneous, since the choice of a federated architecture may be dictated not by technical reasons but rather by organization reasons.

However, the traditional approaches (either based on federation or integration) to the problem of interconnecting heterogeneous databases are not able to deal with the requirements posed by advanced applications such as Office Automation and Computer Integrated Manufacturing (CIM). The reason is that the systems to be integrated include not only traditional DBMSs but also graphic and textual databases. In these systems the semantics of the data is often deeply dependent on the way in which programs manipulate the data and the schema does not exist. The traditional approaches are not able to deal with these latter kinds of databases. In fact the integration approaches used is based on the definition of correspondences between data elements of the database to be integrated (and, possibly, a common global schema in the case of an integrated architecture). This approach is declarative in nature and it is often referenced to as *Structural Mapping*.

In this paper we describe a different approach based on *Operational Mapping*. This approach has a procedural flavor and consists of defining the correspondence between operations at different levels instead of defining correspondences between the data elements. The Operational Mapping is easily supported with the use of an object oriented data model for the implementation of an *abstract view* on top of the local databases. The relationship of the operational mapping technique to an object oriented view is straightforward, since objects are externally known in terms of operations instead than data representation. By defining an object oriented abstract view on top of each local system, the definition of an operational mapping becomes the implementation of a set of operations in an object oriented system. Abstract views are defined using an abstract model which is object oriented.

A system has been developed in order to assess the feasibility of the object-oriented operational mapping approach as part of the ESPRIT Project COMANDOS (COnstruction and Management of Distributed Office Systems). This system, called CIS, is fully described in [Bert 89a] from which this abstract is derived, while a short description of the system has been presented in [Bert 88]. The system has been used to interconnect different application environments, such as relational and graphical databases, and public data banks. [Bert 89a] illustrates an example concerning the integration of a cartographic database implemented as drawings on an AUTOCAD system and of a simple relational database containing information on the transport network of a region.

2. Rationale of the Operational Mapping Approach in CIS

The approach proposed in CIS exploits the idea of separating the definition of operations from their implementation through the notion of *abstract* and *implementation classes*. Basically, an abstract class describes the behavior of an object, i.e. all the operations that the object can respond to and all the properties of the object that may be accessed by another object. An implementation class implements an abstract class by providing

data structures and procedures to construct the abstract class. Implementation classes provide a mechanism that allows the definition of more powerful mappings. In fact, its is possible not only to implement the traditional schema mapping, but also it is possible to provide operation oriented mappings, when the schema mapping cannot be completely applied (or cannot be applied at all). The operation oriented mapping consists in writing explicitly the implementation of the operations associated with the corresponding abstract class as programs local to the data management system to be integrated. Since abstract classes can be accessed only through the operations associated with them, the users of abstract classes have no access to the "representation" of the implementation classes. This provides a complete "encapsulation" of the peculiarities of the data structures used by the local system.

The possibility of having multiple implementation classes corresponding to the same abstract class is a second important point since it provides support for heterogeneity. Therefore given an abstract class, there may be several implementation classes, one for each system storing data that must be accessed through the abstract class. In this way a uniform interface in terms of abstract classes is provided for implementation of integrated distributed applications.

In CIS two levels of operations for manipulating data objects are provided for the integrated applications: *abstract object-at-a-time primitives*, which are based on viewing a class of data objects as a kind of list of objects, and an *associative query language*. When needed, the applications may use the low level primitives and access one object at the time, rather than using the query language. Thereby increased flexibility is provided. The relationships among abstract classes, implementation classes, query language, and applications are summarized in Figure 1. It should be noted that the abstract operations invoked by a program are executed by calling the corresponding implementation operations. A query is processed by being transformed into an access plan, which is a program using abstract operations. An important point is that parameters concerning implementation classes must be provided to the query processor for optimization as part of the integration process..

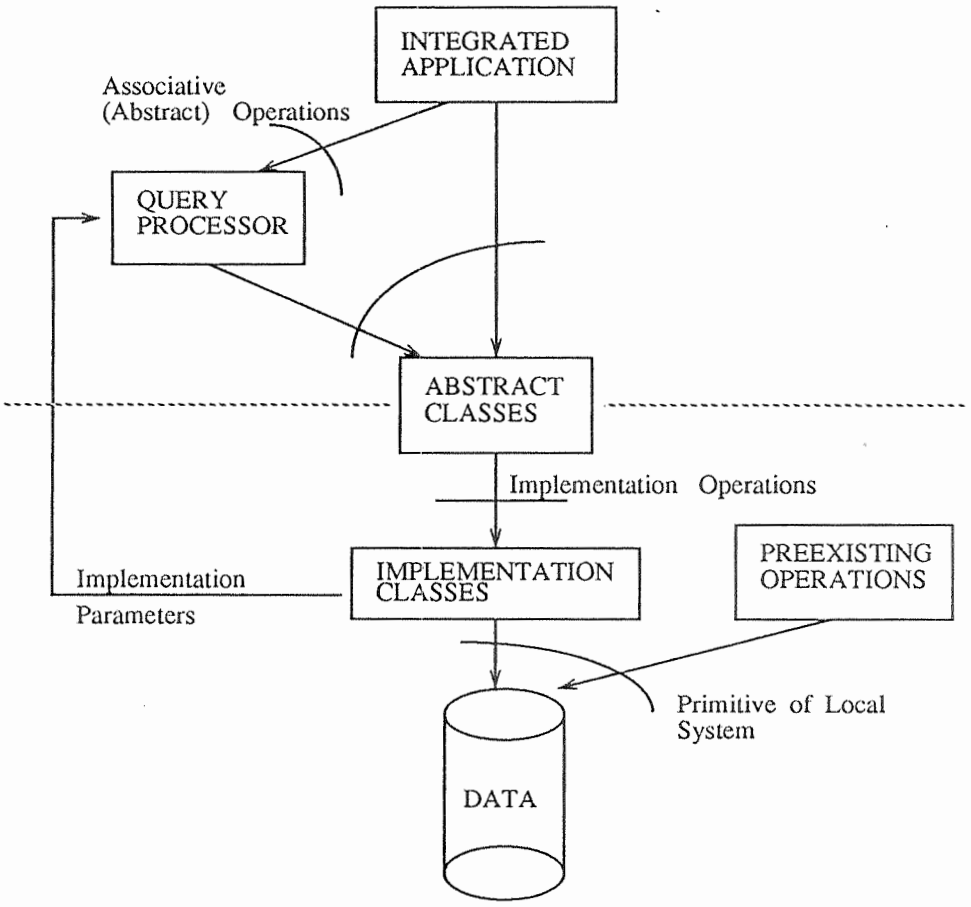
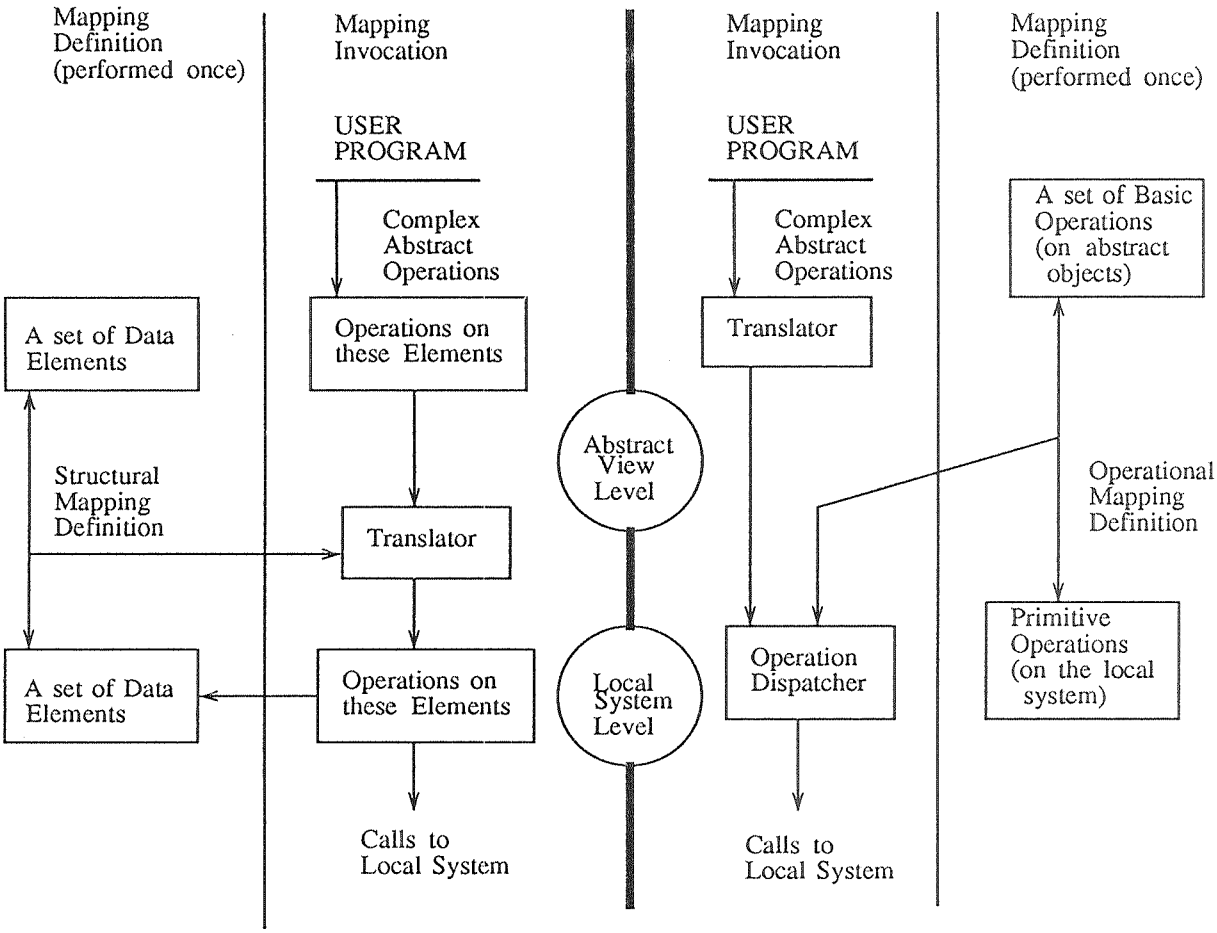


Figure 1. The architecture of CIS

In order to clarify the operational mapping approach, we compare it with the structural mapping. Figure 2.a and 2.b illustrate the basic differences between these mapping techniques. Before analyzing the two figures separately, it is important to note a basic similarity: in both cases the definition of the mapping (which is performed once) is clearly separated from the "Mapping invocation" (which is performed each time that an application program issues a complex abstract operation, such as a query) and in both cases the "Mapping invocation" starts with a program using abstract complex operations and ends with calls to the primitives of the local system. This shows that in *both approaches the goal of the mapping definition is to provide the translation of operations from the abstract, integration level to the local primitives.*

Figure 2.a shows that in order to use the structural mapping an abstract (integration) view is defined as a set of (abstract) data elements; then the structural mapping is defined as a set of correspondences between the data elements of the abstract view and the data elements of the underlying local system. Once that the structural mapping is defined, the integration system supports the automatic translation of operations issued by an integrated application in terms of the primitives operations of the local system. In fact, the mapping is defined only once and all integrated applications use this definition.

Figure 2.b shows the corresponding situation when the operational mapping is used. First, the abstract view is defined as a set of operations on a set of abstract objects. The definition is in terms of an object-oriented data model which is described in [Bert 89a]. Secondly, the operational mapping is defined as the implementation of these abstract operations in terms of the primitive operations of the underlying system. Since the abstract



STRUCTURAL MAPPING

OPERATIONAL MAPPING

Figure 2.a

Figure 2.b

Figure 2. Structural and Operational Mappings

operations must be implemented, it is convenient that they are a possibly minimal set of basic operations, on top of which more powerful abstract operations can be implemented automatically by the integration system. At run-time the abstract operations issued by the integrated application are translated (or interpreted) by the integration system using the basic abstract operations as target, and the implementation of the basic operations is invoked.

In order to define an abstract view as a set of operations it is convenient to use an object-oriented data model, since one of the main characteristics of this model is to view the data through a set of operations, thus encapsulating the internal representation. Building an abstract view in an integration system corresponds to implementing objects in an object-oriented system having the internal representation defined a-priori.

3. CIS Abstract Data Model and Architecture

Data Model

The Abstract Data Model is used in the CIS architecture to define an abstract view. This model is based on the notion of abstract class, which defines the behavior of a set of objects, without defining their internal implementation. An abstract class has a *name* (class name) and a set of *properties*. Each property is defined by a name (property name) followed by the name of its domain. Properties can have as domain both primitive classes (for example integer, boolean, string) or other abstract classes. Properties having as domain abstract classes can be categorized into *independent* and *dependent properties*. The object value of the latter is dependent on the existence on the object of which is part, while this is not true for independent properties [Kim 89]. The model also supports the definition of *variant properties*; that is properties with several different domains in alternative.

The model provides two kinds of operations for handling classes and objects: *generic operations* and *user defined operations*. Generic operations are classified as:

- Operations on classes; these include: insertion of an object in a class, removal of an object from a class, inspection of instances of a class.
- Operations on objects that deal with property manipulation; these include reading and writing of object properties.

User defined operations can be used in a query as functions or predicates provided that the operations have no side effects on the objects stored into the classes. An object-oriented query language has also been defined as part of CIS. The characteristics of this language are discussed in [Bert 89b].

Architecture

CIS supports the creation of integrated applications accessing data managed by heterogeneous and independent application environments residing on different sites of a computer network. These applications may run on different hardware and different operating systems and DBMS, each one with its own conventions on data representation and storage. CIS models this situation with the notion of (logical) nodes, classified into *Client nodes* and *Server nodes*, and of a (logical) communication network between them. The notion of logical node (client or server) in CIS bears no relationship to the notion of physical node as used in communication networks.

A Server is the abstraction of (part of) a preexisting application environment. The role of CIS in a server is to make available to the clients a uniform object-oriented interface on top of the local environment. The CIS component that resides on the server is called *CIS_Server*. A Client is the abstraction of an application based on CIS, accessing the services provided by one or more servers. A client performs an integrated application function using data of different preexisting environments and therefore implements the integration goal of CIS. The CIS component that resides on the client is called *CIS_Client*.

Servers cannot communicate with each other; moreover, each object of the O-O interface must be completely contained in a single server. As a consequence, for CIS there are no relationships between servers except those implemented by clients. A server is therefore an autonomous and independent environment, which groups together a set of objects under a common application semantics. This structure reflects the requirements for loose connection and site autonomy that are relevant for the applications we intend to support. CIS Servers have been designed in order to provide an object-oriented programming environment for the implementation of the abstract classes. The main tasks supported by the environment are:

a) receiving and dispatching messages to the objects (Message Manager)
b) creating objects, supporting object identification and garbage collection (Object Data Manager)
c) defining classes (Dictionary Manager)
d) generating an access plan for queries in terms of object operations (Query Processor).

In CIS implementation classes have been designed taking into account the following facts:
(i) There may exist several implementation classes associated to a given abstract class. This provides a uniform interface toward the applications of data that may have different implementations because of the distributed and heterogeneous nature of the environment.
(ii) Implementation classes must be designed as independent as possible from each other. Information sharing among different implementation classes is avoided in order to obtain implementation classes as a set of "black boxes" communicating only through message exchanges.

An implementation class consists of a set of procedures and some data structures. The data structure used by implementation classes are internal representations which cannot be manipulated in any way by a program operating at abstract level or by procedures belonging to another implementation class. The procedures represent the handles to interact with an implementation class. This concept is quite similar to the programming language concept of abstract type.

An important issue in implementation classes is the identification of objects. The goal of object identification [Khos 86] is to provide the application program with a unique, permanent, unambiguous reference (OID) to an object contained in the local system. This view can be easily supported in situations where the underlying systems provide some kind of database key. However, since CIS must be used to integrate a very broad spectrum of applications, this fact cannot be assumed in general. For example, file systems and graphical systems do not provide this facility. Therefore, the CIS_Server must simulate the existence of OIDs over value-based system. However, the simulation of permanent OIDs is impossible in some situations. Therefore, the restriction has been implemented that the OID returned by a Server to a Client in response to some operation is valid only until the end of the current session. A consequence of this restriction is that OIDs cannot be stored into files and cannot be passed among Clients. However, this restriction makes the implementation of OIDs possible even in environments which do not support the identification of objects directly. The way in which the ODM supports this task is described in [Bert 89a].

4. Conclusions and Future Work

In this paper, we have described an approach to the integration of heterogeneous systems, based on the object-oriented paradigm. The real feasibility of this approach has been tested by integrating several types of data repositories; one of these experiences, involving graphical data, is described in [Bert 89a]. In this experiment, the data integrated were managed by an independent package (AUTOCAD) and the specific semantics of these data was embedded in the application logic. The integration problem of the kind described in this paper and in [Bert 89a] is crucial to the development of a broad range of new applications, because large information systems are more and more built by integrating different packages, applications, data repositories.

CIS has demonstrated that the proposed approach is feasible. However, further research is needed to design tools that reduce the effort required for building an object-oriented interface and to determine which cases can be dealt with this approach and which cases cannot. Another important research issue concerns query processing and in particular query optimization. In our environment these tasks are complicated by the hiding of object internal structure (because of the encapsulation principle of the O-O paradigm). However the knowledge about object internal structures is necessary to perform effective query optimization. Furthermore the query processor must be flexible enough to manage all the possible access structures and secondary storage organizations provided by the underlying systems.

Since the query processor must adapt to a number of subsystems, this requires the knowledge on access strategies provided by the underlying system be represented as input parameters rather than being embedded in

the query optimizer code. The Class Implementor should then provide those data as part of the integration. Providing this flexibility implies the usage of an *extensible optimizer* [Lohm 87]. Although some experiences on this field are under development, the problem is still an open research issue. Therefore for the first system prototype (now under development) we have used few heuristics rather than providing a real query optimization. However, we think that extensibility techniques could be appropriate to solve the problem of query optimization in our environment. We plan to deeply investigate this issue in the near future.

Bibliography

[Bati 86] C.Batini, M.Lenzerini, and S.B.Navathe, "A comparative analysis of methodologies for DB schema integration" *ACM Computing Surveys*, Vol.18, No.4, Dec.1986.

[Bert 88a] E.Bertino, F.Rabitti, S.Gibbs, "Query processing in a multimedia document system", *ACM Trans. on Office Information Systems*, Vol. 6, No. 1, Jan. 1988.

[Bert 88b] E.Bertino, R.Gagliardi, M.Negri, G.Pelagatti, L.Sbattella, "The COMANDOS integration system: an object-oriented approach to the interconnection of heterogeneous applications", *Proc. of the 2nd International Workshop on Object-Oriented Database Systems,* Bad Munster am Stein-Ebernburg, FRG, Sept.1988.

[Bert 89a] E.Bertino, M.Negri, G.Pelagatti, L.Sbattella, "Integration of heterogeneous database applications through an object-oriented interface", *Information Systems,*Vol.14, No.5, 1989.

[Bert 89b] E.Bertino, M.Negri, G.Pelagatti, L.Sbattella, "Object-oriented query languages: the notion and the issues", submitted for publication, 1989.

[Conn 88] T.Connors, P.Lyngbaek, "Providing uniform access to heterogeneous information bases", *Proc. of the 2nd International Workshop on Object-Oriented Database Systems,* Bad Munster am Stein-Ebernburg, FRG, Sept.1988.

[Daya 84] U.Dayal, H.Hwang, "View definition and generalization for database integration in a multibase system", *IEEE Trans. on Sotfware Engineering*, Vol.SE-10, No.6, Nov.1984.

[Land 82] T.A.Landers, R.L.Rosenberg, "An overview of Multibase", in *Distributed Data Bases,* H.S. Schneider, ed., North-Holland, New York, 1982.

[Litw 85] W.Litwin, "An overview of the multidatabase system MRDSM", *ACM Nat'l Conf.*, Denver, Oct. 1985.

[Litw 86] W.Litwin, A.Abdellatif, "Multidatabase interoperability", *IEEE Computer*, Dec. 1986.

[Lohm 87] G.Lohman, "Grammar-like functional rules for representing query optimization alternatives", RJ 5992, IBM Almaden Research Center, San Jose, CA 95120, Dec. 1987.

[Khos 86] S. Khoshafian, G.Copeland, "Object Identity", *Proc. 1st Intl. Conf. on Object-Oriented Programming Systems, Languages, and Applications,* Portland, Oregon, Oct. 1986.

[Kim 89] W.Kim, E.Bertino, J.Garza, "Composite Objects Revisited", *Proc. ACM-SIGMOD Conf. on Management of Data*, Portland (Oregon), May-June 1989, also MCC Technical Report No. ACA-ST-387-88, Nov. 1988.

[Motr 87] A.Motro, "Superviews: virtual integration of multiple databases", *IEEE Trans. on Sotfware Engineering*, Vol.SE-13, No.7, July 1987.

[Salt 83] G.Salton, and MJ.McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[Ston 77] M.Stonebraker, E.Neuhold, "A distributed database version of INGRES", in *Proc. of Berkeley Workshop on Distributed Data Management Systems*, Berkeley, California, May 1977.

[Will 81] R. Williams, et Al., "R*: an overview of the architecture", Technical Report RJ3325, IBM, Feb. 1981.