LBL-29576
UC-405

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Information and Computing Sciences Division

Data Management in Heterogeneous Environments and its Implications to CEDR

V.M. Markowitz

October 1990

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

DATA MANAGEMENT IN HETEROGENEOUS ENVIRONMENTS AND ITS IMPLICATIONS TO CEDR*

Victor M. Markowitz†

Data Management Group
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

October 27, 1990

* This work was supported by the Office of Health and Environmental Research Program and the Applied Mathematical Sciences Research Program, of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.

† Author's E-mail address: *V_Markowitz@lbl.gov* Office phone number:(415) 486-6835

CONTENTS

Executive Summary i
1. Introduction 1
2. Semantic Heterogeneity 3
2.1 Naming Diversity 3
2.2 Domain Diversity 3
2.3 Structural Diversity 3
2.4 Behavioral Diversity 4
2.5 Incomplete Information 4
2.6 Object Identification 4
3. Heterogeneous Database Integration 5
3.1 Physical Integration of Heterogeneous Databases 5
3.2 Virtual Integration of Heterogeneous Databases 6
4. Schema Integration 8
4.1 Schema Integration Stages 8
4.2 Global and Local Schema Integration 10
4.3 Limitations 12
5.Data Exchange and Sharing in a Multidatabase System 13
5.1 Data Exchange in a Multidatabase System 13
5.2 Data Sharing in a Multidatabase System 15
6. Conclusions and Recommendations 17
References 22
A. Semantic Heterogeneity in CEDR 24

EXECUTIVE SUMMARY

OVERVIEW

We investigate in this report the issue of data management across multiple pre-existing databases characterized by various degrees of *heterogeneity*. Different approaches to the problem of data management in heterogeneous environments are reviewed and their advantages and disadvantages are discussed. We examine in some detail the problem of schema integration involved in these approaches. We illustrate different aspects of data management in heterogeneous environments with examples from the *Comprehensive Epidemiological Data Resource* (CEDR) project, and conclude the report with recommendations for CEDR. These recommendations are summarized below.

RECOMMENDATIONS

Data management in heterogeneous environments involves *integrating* heterogeneous databases in order to provide users with a unified view of the data. The two main approaches to database integration are (i) *physical database integration*, which implies physically replacing all existing databases with a central database; and (ii) *virtual database integration*, where existing databases and applications are preserved, but users are provided with a unified (*virtual*) view of the data. The main problem of the first approach concerns the complexity and high cost of the integration process, which involves in addition to schema integration, integrating (i.e. converting and merging) the associated data and adapting the applications that are based on the existing databases to the central database. Since such a process is beyond the objective of CEDR of providing a mechanism for accessing epidemiologic data we recommend that:

Recommendation 1 : Physical database integration should not be adopted for CEDR.

The main problem of the second approach approach concerns the complexity of the mechanisms needed to support the interoperability of the heterogeneous databases. Since many of these mechanisms are not available commercially, and because developing all these mechanisms would require a great deal of time and effort, we recommend that:

Recommendation 2 : Virtual database integration should not be adopted for CEDR.

Following an analysis of the characteristics of the CEDR application, we recommend that:

Recommendation 3 : A compromise between the physical and virtual database integration approaches should be adopted for CEDR.

This approach follows the virtual database integration approach by preserving the existing databases and applications, but compensates the lack of interoperability mechanisms by physically integrating selected data subsets extracted from existing databases.

We outline below two strategies for the implementation of the compromise integration approach recommended above, and discuss briefly the reasons for preferring one of these strategies over the other.

Strategy 1 : Standard-driven: conversion at local sites, integration at the CEDR site.

This strategy involves (a) developing a standard for the integrated CEDR database, including data elements (variables), formats, and units; (b) requiring each site to extract and convert the data according to the standard; and (c) verifying and merging the converted data at the CEDR site; if the verification fails, the sites involved are asked to redo the conversion. Note that the current strategy of requiring the sites to generate files corresponding to the IARC standard, conforms to strategy 1.

Strategy 2 : Standard-independent: conversion and integration at the CEDR site.

This strategy involves (a) requiring the sites to send to the CEDR site all datasets (e.g. analysis files) that may be of interest to the CEDR user community, regardless of any specific standard; (b) describing each dataset using an agreed upon exchange format; (c) verifying, converting, and integrating the datasets at the CEDR site.

Strategy 2 is more flexible than strategy 1 by promoting the incorporation of potentially useful data in the CEDR database independent of any standardization. While strategy 1 depends on the sending sites to carry out the data conversions according to a standard, strategy 2 shifts the responsibility for conversion to the CEDR site, thus eliminating the need of involving the sites in correcting imprecise conversions. Strategy 1 implies a simpler integration process at the CEDR site, but it does not eliminate the need for dealing with all the aspects of integration during the standardization process itself. Note that strategy 2 does not eliminate standardization, but rather removes it from being on the critical path between the various sites and the CEDR site. Thus, standardization can still determine the definition of *views* of the CEDR database. Another advantage of strategy 2 is that changes to the standards do not require another round of conversions by the sites, since conversions are handled at the CEDR site.

The advantages of strategy 2, namely its flexibility in terms of the content of the CEDR database, its non-reliance on multiple sites to perform conversions, and its independence from the standardization processes, lead us to recommend that:

Recommendation 4 : Strategy 2 outlined above should be adopted for CEDR.

We conclude with directions for future work. We recommend to further develop the details involved in the strategy 2 outlined above. This will entail (a) developing a schema integration methodology, (b) investigating data conversion strategies, (c) exploring mechanisms for maintaining the consistency of replicated data in CEDR, and (d) develop conventions for describing the datasets.

1. INTRODUCTION

Database management systems (DBMS) were proposed in order to provide shared access to heterogeneous collections of data created by multiple autonomous applications [LITW89]. Database management systems have the following main characteristics [ULLM88]:

(i) the ability to access efficiently large amounts of data;

(ii) support for a *data model* that provides the abstraction mechanisms for modeling data;

(iii) support for *high-level languages* that allow users to define, access, and manipulate data concisely;

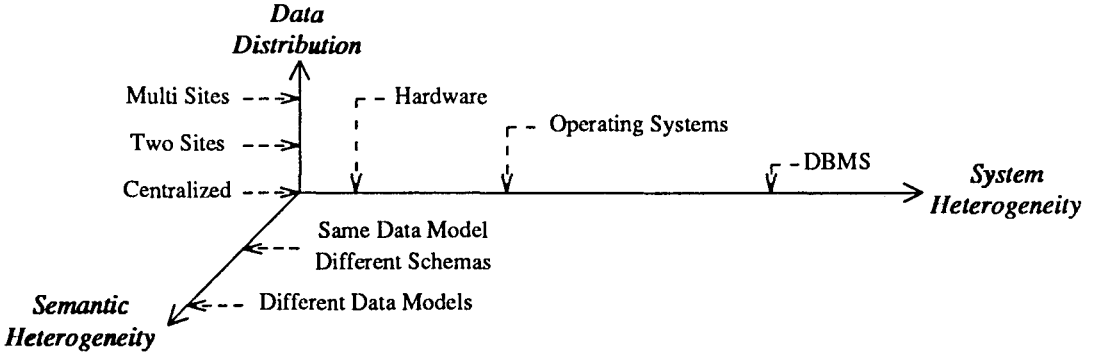
(iv) *transaction management*, the capability to provide correct, concurrent access to the database by many users at once; and

(v) *access control*, the ability to limit access to data by unauthorized users and the ability of checking the validity and integrity of data.

These characteristics distinguish a DBMS from other systems that have the capability of managing persistent data, such as file systems. Thus, file systems do not have a high-level language for updating or accessing data, do not provide transaction management, and do not have mechanisms for checking the validity and integrity of data.

The employment of a DBMS implies organizing the data in a central database shared by multiple applications, rather than allowing every application to have its own data organization. Thus, the database provides a *centralized* and *homogeneous* view of the data for the various applications.

In this report we address the problem of managing data in multiple pre-existing databases characterized by various degrees of *heterogeneity*. The following diagram captures the three main heterogeneity aspects characterizing such databases, namely data distribution, system heterogeneity, and semantic heterogeneity:



Various aspects of semantic heterogeneity are reviewed in section 2.

The *Comprehensive Epidemiological Data Resource* (CEDR) project at LBL is an example of a data management application developed in a heterogeneous environment. The CEDR data is currently distributed among three main sites (LANL, ORNL, PNL). These sites employ different hardware (IBM, DEC) under different operating systems, and use different DBMSs (DATATRIEVE, ORACLE). Although the DBMSs mentioned above are all *relational* systems, that is, are based on the same (relational) data model, the data in these databases is structured differently (have different *schemas*); moreover, the common part in these databases (e.g. the same variables) are viewed differently, that is, the corresponding domains are different (see the appendix for more details).

The management of data in multiple heterogeneous databases can be achieved in two main ways:

(1) by physically integrating the pre-existing databases into a new unified database, or

(2) by virtually integrating, and without altering, the pre-existing databases.

The advantages and disadvantages of these two approaches are discussed in section 3. Both approaches involve integrating the schemas of the pre-existing databases. The issue of schema integration is discussed in section 4. In section 5 we examine the mechanisms that are required for supporting data exchange and data sharing between heterogeneous databases. We conclude this report with specific recommendations for CEDR.

2. SEMANTIC HETEROGENEITY

Semantic heterogeneity is an expression of the diversity of the data modeling process. We review below several aspects of semantic heterogeneity encountered in heterogeneous database environments.

2.1 Naming Diversity.

Database objects are assigned *names* according to certain algorithms, standards (e.g. nomenclatures), and/or natural language connotations. However, certain natural language and nomenclature words, called *homonyms*, can designate multiple concepts, while certain concepts can be designated by multiple words, called *synonyms*. In a database homonyms may cause *naming conflicts*, while synonyms may cause *naming inconsistencies*.

For example, CLIENT and CUSTOMER designate the same concept, that is, are synonyms. Codes for diseases are an example for nomenclature synonyms. A common example of a homonym encountered in the data sources for CEDR is DATE which can be used in order to refer to *year*, to the combination of *year, month, day*, or other combinations of time parameters. Another example of a homonym in CEDR is SOCIO-ECONOMIC-STATUS which is based on *job classification* in some data sources, and *years of education* in other data sources.

2.2 Domain Diversity.

Databases store information on objects, such as values for object attributes (properties). Attribute values are taken from *domains*, and domains can be associated with *units* and *formats*. For example, the unit for a domain of SPEED values can be *miles/hour*, and the format for a domain of DATE values can be *Month/Day/Year* as a string of 8 characters. Units and formats, however, are not unique. Thus, *kilometers/hour* can be also a unit for SPEED, and *Day/Month/Year* can be also a format for DATE.

2.3 Structural Diversity.

Objects are represented in databases in various ways. An object representation depends both on how the object is perceived, and on the data model (i.e. language) employed to describe the object. Thus, different data models (e.g. *relational*, *Extended Entity-Relationship* (EER)) provide different constructs for specifying object structures (e.g. tables in the relational model, object-sets and generalization in the EER model), and constraints (e.g. functional and inclusion dependencies in the relational model, cardinality constraints in the EER model). Moreover, different users may perceive differently a certain real-world object even when they employ the same data model. For example, a FACILITY can be perceived by different users either as an object or an attribute associated with another object (e.g. WORKER).

2.4 Behavioral Diversity.

Objects can differ not only in the way they are represented in databases, but also in their associated behavioral rules (sometimes called *policies*). For example, suppose that the relationship between WORKER and FACILITY objects is represented in two schemas, so that in one schema a FACILITY is allowed to exist without any WORKER, while in the other schema a FACILITY is not allowed to exist without a WORKER. These schemas differ in their behavioral rules, with the rules in the second schema implying that the deletion of the last EMPLOYEE associated with a given FACILITY is accompanied by the deletion of that FACILITY.

2.5 Incomplete Information.

Incomplete (missing) information is represented in databases using special purpose *null* values. However, a null value can have different meanings, such as *unknown*, *unavailable*, *inapplicable*, etc. Thus, 14 different meanings for null values are listed in [ANSI75]. These different meanings are particularly important when null values are manipulated. For example, unlike an unknown value, an *inapplicable* value cannot be changed into a non-null value.

Nulls are not always necessary, but can be the result of selecting a certain representation for an object structure. For example, suppose that a subset of the set of WORKERS consists of DECEASED WORKERS, and that this subset is associated with attribute CAUSE-OF-DEATH. The value of CAUSE-OF-DEATH can be unknown for some DECEASED WORKERS, but is always applicable for all DECEASED WORKERS. If attribute CAUSE-OF-DEATH, however, is associated with the set of WORKERS, then for a worker who is not a deceased worker, the value for CAUSE-OF-DEATH is inapplicable.

2.6 Object Identification.

There are various ways of identifying objects in databases (e.g. see [KHOS86]). For example, in relational databases an object can be identified by a subset of its (*primary-key*) attribute values, or using system-controlled values, called *surrogates*; surrogates carry no information, and users can cause their deletion or generation, but cannot update them [KHOS86].

Object identification is not uniform. For example, in different databases a person can be identified using (e.g. as primary-key) the social security number, or an employee number, or the combination of the first and last names. Similarly, if surrogates are used as object identifiers, it is likely that an object will be represented by different surrogates in different databases. For example, in different CEDR data sources WORKERS are identified using so-called *pseudo-identifiers* (which are similar to surrogates), but in different data sources common WORKERS are identified by different pseudo-identifiers.

3. HETEROGENEOUS DATABASE INTEGRATION

Data management in heterogeneous databases requires having a uniform (*integrated*) view of the data in these databases. The main approaches to achieve such an integration are briefly discussed below.

3.1 Physical Integration of Heterogeneous Databases.

The first approach to heterogeneous database integration does not allow system and semantic heterogeneity. The pre-existing databases are *integrated* into a common *global* database, that can be either *centralized* or *distributed*. In both cases the global database is *homogeneous* and physically replaces the pre-existing databases. *Views* that correspond to the pre-existing databases can be defined over the global database. The schema architecture for such a global homogeneous database is shown in figure 3.1.

The main advantages of this approach are the following:

1. Users need to know only one global schema, and can access the data in a uniform way.
2. The global database involves only a few, relatively simple, schema and query translation facilities. Such facilities are supported, for example, by relational DBMSs.
3. No new mechanisms are needed for query processing and for enforcing integrity constraints over the global database.

The main drawbacks of this approach are the following:

1. Data and schemas of pre-existing databases must be integrated. Integration requires agreement

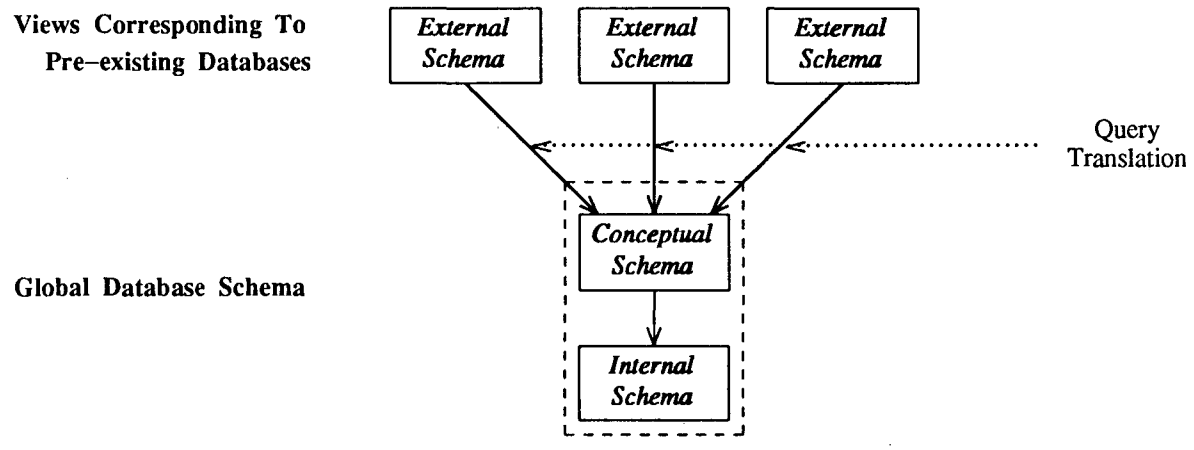


Figure 3.1 Schema Architecture for a Physically Integrated Database.

among users on common (global) names, structures, values, and policy. Such an agreement is difficult to attain, and the result can be unsatisfactory to some (possibly all) users.

2. All existing applications (e.g. queries, updates, etc) must be converted in order to comply with the new global elements. This conversion process is usually very expensive, and not always feasible. The view capability provided by RDBMS is intended to decrease the impact of replacing the local databases with a global database, but is very limited in handling updates and conversions (e.g. formats, units).
3. The manipulation of a large, centralized or distributed, database is inherently more complex than that of smaller local databases; in particular, updating and database reorganization becomes more complex and requires controlling their side effects on local databases or views.

3.2 Virtual Integration of Heterogeneous Databases.

The second approach to heterogeneous database integration allows both system and semantic heterogeneity. A virtual schema allows accessing the local (pre-existing) databases, while the local databases are not affected, thus ensuring their *autonomy*. The schema architecture of a virtually integrated heterogeneous database is shown in figure 3.2. Three types of autonomy are pursued by this approach [REPO89]: (i) *design autonomy* refers to the capability of the local databases to have their own data model and schemas; (ii) *communication autonomy* refers to the capability of the local databases to decide with what other databases to communicate and what information to exchange; (iii) *execution autonomy* refers to the capability of the local databases to decide how and when to execute requests from other

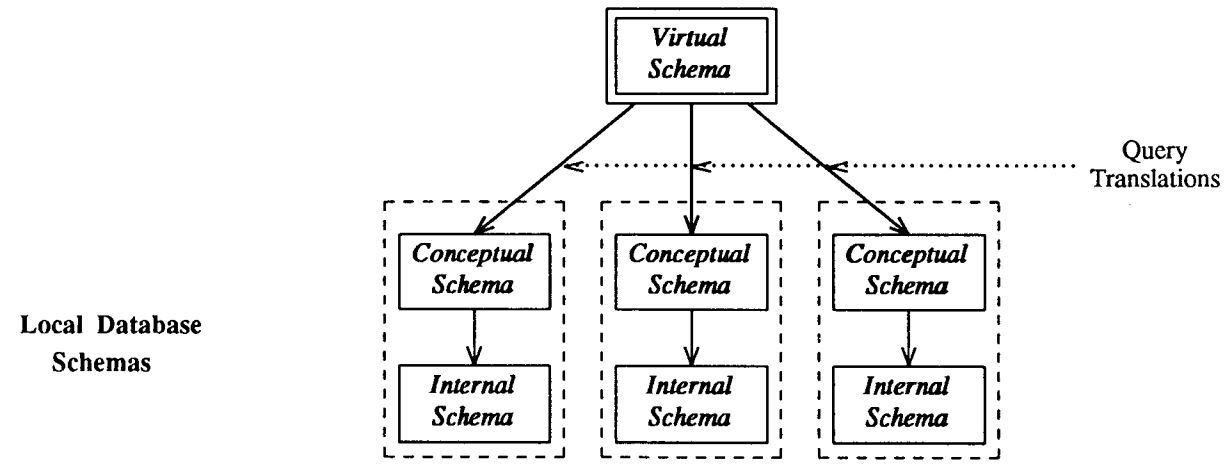


Figure 3.2 Schema Architecture for a Virtually Integrated Database.

databases.

The main advantages of this approach are the following:

1. Users of the local databases preserve their views of the data, and continue to use the (already known) languages provided by the local DBMSs.
2. Existing applications on local databases are not affected by the integration.
3. Updates in local databases do not affect other local databases.

The main problems raised by this approach are the following:

1. Users need to know more than one schema and more than one data model, because in most cases the virtual schema is described with a different data model than the local schema.
2. Accessing local databases via the virtual schema requires solving semantic differences and conflicts between the local schemas. This problem, which is also encountered in the physical database integration, is complex.
3. The multiplicity (redundancy) of semantically identical data values in the local databases cannot be reduced, and is hard to maintain. Thus, new inter-database dependencies ([LITW89], [MARK90b]) must be introduced and enforced.
4. Numerous and complex schema and query translation mechanisms are needed.
5. Transaction and query processing require some form of cooperation among local databases. Transaction and query processing (in particular query optimization) in heterogeneous environments are especially difficult when execution autonomy is pursued.

4. SCHEMA INTEGRATION

Both physical and virtual integration of heterogeneous databases discussed in the previous section involve integrating database schemas. In this section we discuss the various stages of schema integration, and two different approaches to schema integration.

4.1 Schema Integration Stages.

A comparative study of schema integration methodologies is provided by [BATI86], where four main stages (activities) are identified for the process of schema integration:

1. *Preintegration* refers to the preliminary stage of integration, in which (i) a common data model is selected; and (ii) the local database schemas are described (converted) using this common data model.
2. The *comparison* of the schemas involved in integration is carried out in order to detect conflicts and inconsistencies regarding names, structures, etc.
3. The *alignment* (or *conformation*) of the schemas involved in the integration resolves the conflicts detected in the previous stage, in order to achieve pairwise compatibility. Subsequently, the *similarity* of object classes is established in preparation for merging the schemas.
4. *Merging* of compatible schemas into a global schema.
5. *Restructuring* the global schema in order to eliminate redundancies.

The schema integration methodologies differ in the way they implement these stages. A comprehensive comparison of twelve integration methodologies is provided by [BATI86].

The common data model used in the *preintegration* stage is usually an object-oriented data model, such as the *Entity-Relationship* (ER) model [NAVA86b] or the *functional* model [MOTR87]. Converting schemas of local databases using such a common data model is a non-trivial process especially when the data model of the local database and the common data model have different constructs. Thus, relational schemas are not always mappable into object-oriented schemas [MARK90a].

In order to keep track of name correspondences, in the *alignment* stage a data dictionary (e.g. see [NAVA86a]) or thesaurus (e.g. see [NISO90]) facility can be used. Domain mismatches are solved by specifying *virtual attributes* and *domain mappings* (conversions) from the real attributes to the virtual attributes [DEMI89]. Resolving structural differences usually involve schema (restructuring) transformations (e.g. see [MOTR87], [MARK88]).

A simple example of integration is illustrated in figure 4.1. The IARC files provided by two different CEDR sites, PNL and LANL, include data about the DATE OF BIRTH and SOCIO-ECONOMIC STATUS of WORKERS; WORKERS are identified by a pseudo-identifier denoted ID. In the *preintegration* stage the EER model is selected as the common data model for representing these data, as shown in figure 4.1(i). The *comparison* of the two schemas shown in figure 4.1(i) reveals two conflicts:

(i) A name conflict for SOCIO-ECONOMIC STATUS, which has two different meanings in these schemas, namely *Job Classification* for the PNL schema and *Years of Education* for the LANL schema, respectively. Accordingly, these schemas are *aligned*, that is, restructured in order to become compatible. The restructuring involves (a) converting the SOCIO-ECONOMIC STATUS from an attribute of WORKERS into an entity-set associated by relationship-set HAS with WORKERS, and then (b) splitting the SOCIO-ECONOMIC STATUS entity-set into two entity-sets, SOCIO-ECONOMIC TYPE

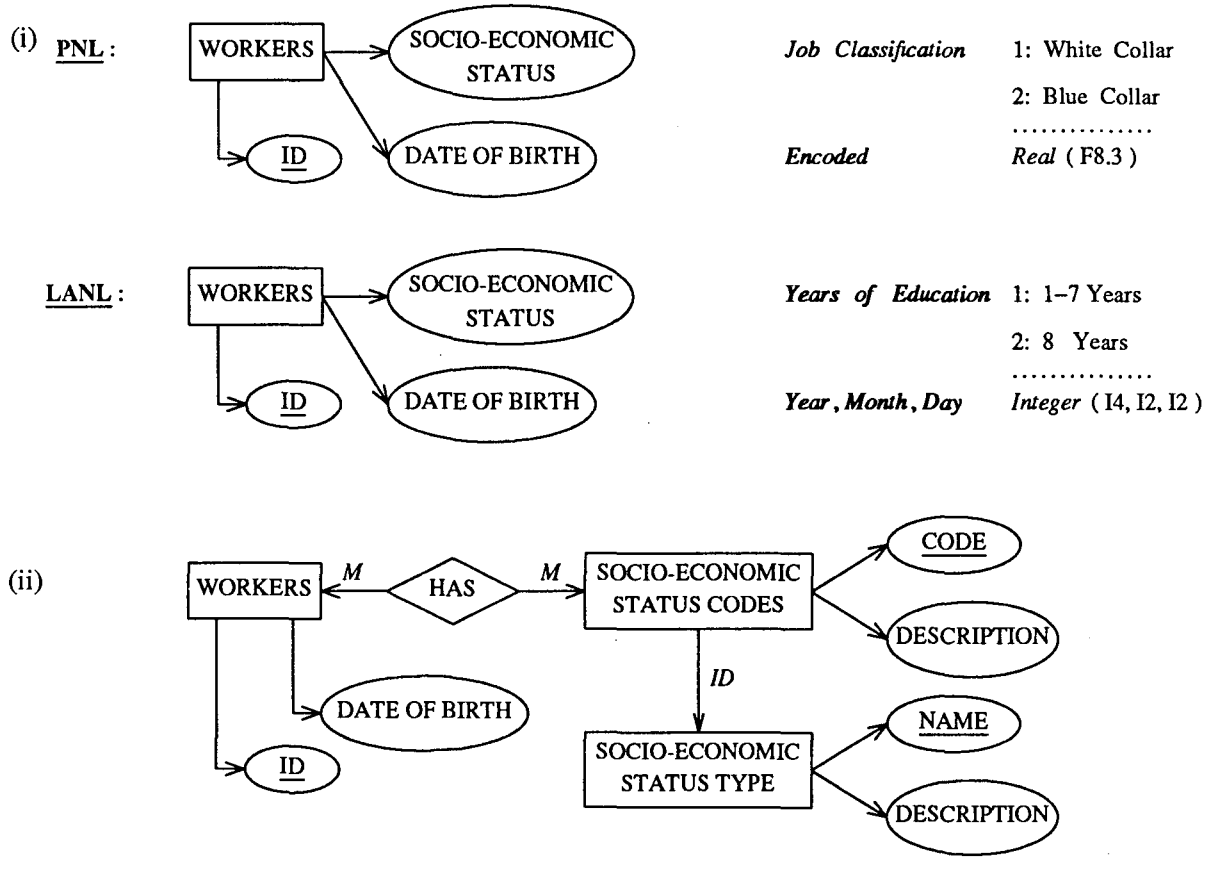


Figure 4.1 An Example of Schema Integration.

containing the various types of socio-economic data (e.g. *Job Classification*, *Years of Education*), and SOCIO-ECONOMIC CODES containing the codes for every type of socio-economic data (e.g. 1, 2, ... for *Job Classification*, etc.).

(ii) A domain conflict for DATE OF BIRTH, which has two different formats: a real number (F8.3) format for encoded dates (the encoding formula is [Year + (Month-1)/ 12 + (Days-0.5)/ 365]) for the first schema, and three integer numbers (for year, month and day, respectively) for the second schema. In the *alignment* stage a new (*virtual*) attribute is specified, and associated with a domain of type *datetime* (this type, which is provided by all RDBMSs, allows using special date functions, such as comparisons). *Domain mappings* from the domains associated with the original DATE OF BIRTH attributes to the domain of the virtual attribute are also defined in this stage.

Finally, the two (now compatible) schemas are merged into the integrated schema shown in figure 4.1(ii). For this simple example of integration no restructuring is needed in order to eliminate redundancies.

4.2 Global and Local Schema Integration.

Schema integration can be performed either *globally*, by creating one integrated (global) schema for all the local databases, or *locally*, by creating integrated schemas for every local database. While virtual database integration can involve either global or local schema integration, physical database integration can involve only global schema integration.

Global schema integration creates a *global* schema from the local schemas, as illustrated in figure 4.2(i). In the *preintegration* stage local schemas are converted into equivalent schemas, called *export* schemas, specified using a common data model. Subsequently, export schemas are integrated into a global schema, specified using the same common data model.

The second approach to schema integration is more flexible than the first approach by allowing local schema integrations. This approach, illustrated in figure 4.2(ii), follows the architecture described in [HEIM85]. Every local database defines a subset of its conceptual schema as its *export* schema; the export schema specifies the information that a local database allows to be accessed by external users. Conversely, the *import* schema in a local database specifies the information that a local database needs from other local databases, described in the language (data model) of the local database. Export schemas are organized (federated) into a *federated* schema that can be used by external users to access the local databases. The federated schema can range from a loose collection of export schemas to a fully integrated schema, such as the global schema discussed above. Every local database takes care of locally integrating its conceptual and import schemas, thus creating a *personalized global view* of the database [FANK88];

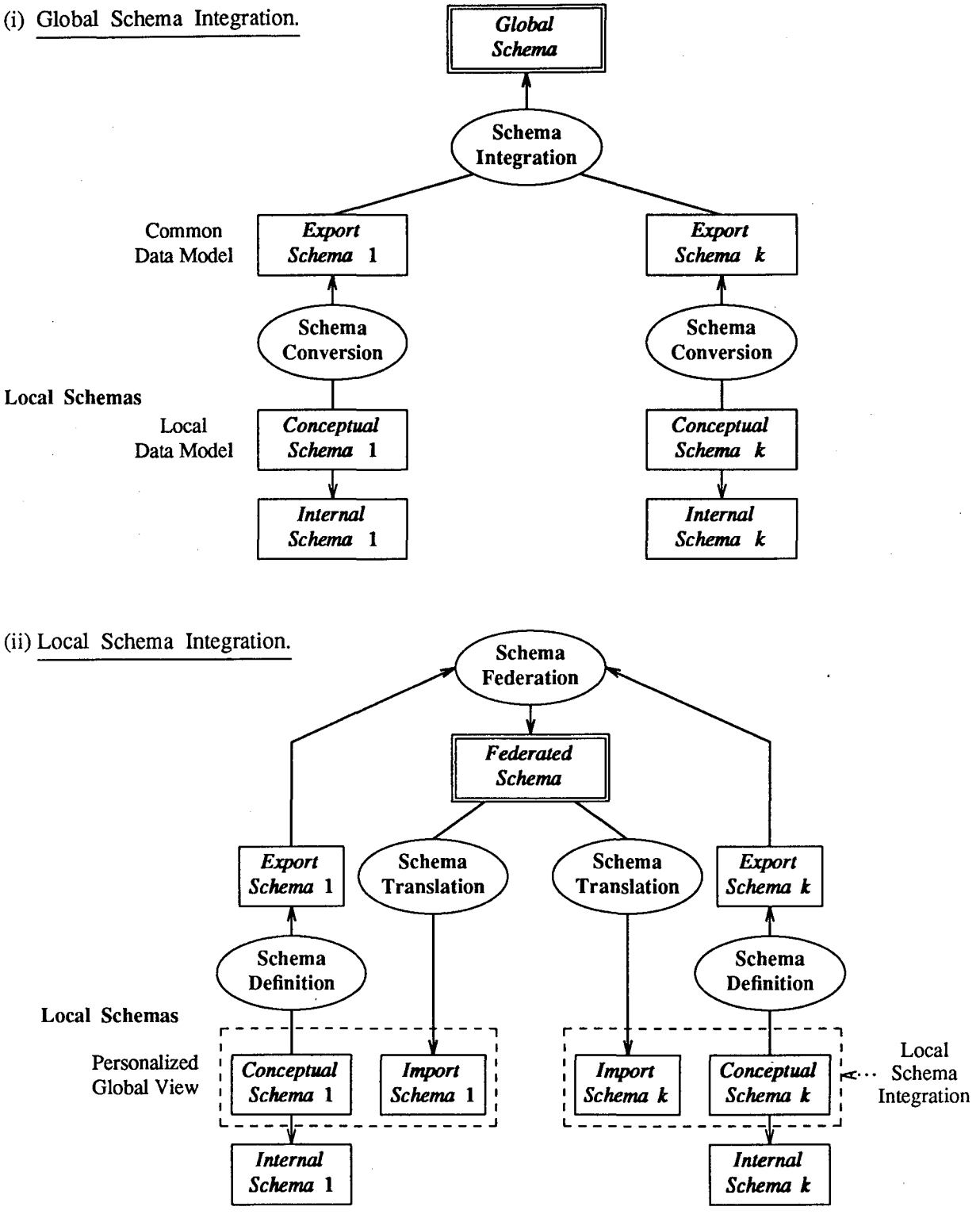


Figure 4.2 Global and Local Schema Integration.

alternatively, local databases can be provided with a *multidatabase language* [LITW87] that supports the joint manipulation of data in different databases. In the later case the user is not provided with a static global view of the multiple databases, but rather has to create dynamically such a view using the data definition capabilities of the multidatabase language.

4.3 Limitations.

The schema integration methodologies described in the literature and surveyed in [BATI86] have several problems which are briefly discussed below.

Several aspects of schema integration, such as resolving conflicts, are addressed only superficially. Thus, most methodologies use only *renaming* in order to resolve naming conflicts and inconsistencies. Clearly, renaming is insufficient for schema integration involved in virtual database integration where a thesaurus mechanism should be employed.

Almost all schema integration methodologies (with the exception of a few, very restricted, methodologies) provide general guidelines, rather than an algorithmic specification of the integration process. In particular, there is no proof of *completeness* for the schema transformation operations involved in the *alignment* stage, that is, it is not clear whether these operations are sufficient for resolving all types of conflict.

One of the most controversial aspects of schema integration methodologies concerns establishing the *similarity* of object classes in the *alignment* stage. Most methodologies assume that this process is not automated and is entirely manual. Conversely, methodologies such as that of [NAVA86b] attempt to automate at least in part this process by (i) first determining the relationships between pairs of attributes, and then (ii) using *resemblance functions* for detecting classes of similar objects [LARS89]. As noted in [SHET89], the process of determining attribute relationships is hard to automate. Thus, [SHET89] shows that the technique used in [LARS89] for determining attribute relationships is ad-hoc and inaccurate.

There are only a few reported attempts to implement schema integration tools (e.g. [SHET88]). The goal of these tools was to demonstrate the feasibility of the underlying methodologies, rather than being complete implementations. Accordingly, these tools are only partial experimental implementations. Furthermore, these tools have not been employed in real projects requiring database integrations.

5. Data Exchange and Sharing in a Multidatabase System

In section 3 we have discussed two approaches to heterogeneous database integration, namely physical and virtual database integration. While a physically integrated database does not need any new mechanisms, a virtually integrated database must be complemented with mechanisms supporting the exchange and sharing of data in the heterogeneous *multidatabase* system. In this section we briefly discuss these mechanisms.

We assume that every (local) database is associated with a metadatabase (or data dictionary) containing information describing the database, such as the database schema, integrity constraints, domains, etc. As a result of the schema integration process, a *meta-multidatabase* is created for the heterogeneous multidatabase system. This meta-multidatabase (called *federated dictionary* in [HEIM85]) incorporates a *thesaurus*, and contains information describing the correspondences and mappings between the databases of the multidatabase system. If the schema integration is local then every database has its own meta-multidatabase reflecting its *personalized global view* of the multidatabase, as discussed in section 4.2. We assume below that the schema integration is global so that all the databases share a common meta-multidatabase.

5.1 Data Exchange in a Multidatabase System.

Databases in a multidatabase system must be provided with mechanisms supporting data exchange with other (foreign) databases in the multidatabase. A database needs a *multidatabase user interface* (see figure 5.1) that supports the specification of queries and browsing operations on the multidatabase, and that is capable of displaying the results of these operations in the specific (local) way of the database.

The data exchange between a (local) database and other (foreign) databases is mediated by a *data exchange controller* (see figure 5.1). This mechanism supports the execution of queries across multiple heterogeneous databases by

(i) 1. *decomposing* queries that involve multiple databases into single-database subqueries,

2. *translating* these subqueries into queries accepted by the foreign databases, and

3. *dispatching* the queries to the foreign databases;

(ii) 1. *gather* the query results from each database;

2. *convert* the results into a common format, and

3. *synthesize* the results into a unified result.

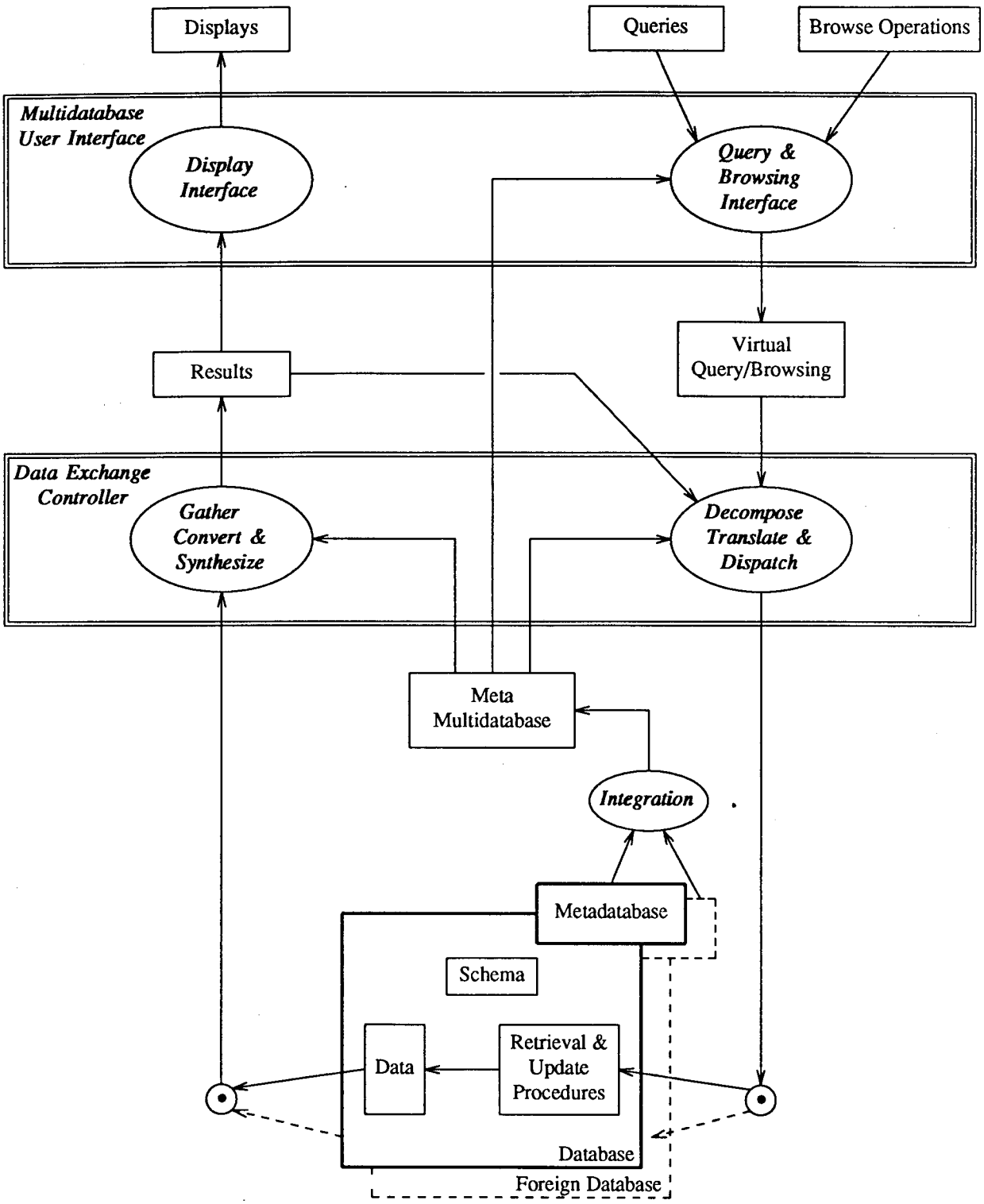


Figure 5.1 An Architecture for a Multidatabase System.

A more complex architecture for centralized (rather than local) data exchange mechanisms is sketched in [NAVA89]. Related architectures are discussed in [HEIM85].

Data exchange in a multidatabase system requires some form of cooperation for processing transactions and queries. Traditional (i.e. centralized) transaction and query processing conflicts with the *execution autonomy* that allows databases in a multidatabase system to decide how and when to execute requests from other databases. Multidatabase systems need new transaction processing (e.g. concurrency control) mechanisms and query processing (e.g. optimization) techniques. Research in these areas is at an early stage [REPO89] (see also [BREI90]).

5.2 Data Sharing in a Multidatabase System.

Databases contain data about *objects* and object *connections*. The databases in a multidatabase preserve their *autonomy* in structuring and manipulating their data. Thus, objects may be represented and identified differently in different databases of a multidatabase. An architecture for identifying objects across relational databases involved in a multidatabase is presented in [MARK90b]. This architecture, illustrated in figure 5.2, involves extending every database of the multidatabase with two relations:

1. The purpose of the *local-objects* relation is to list the identifiers of all the objects represented in a database r. Thus, the local-objects relation for a database r includes all the key values that appear in

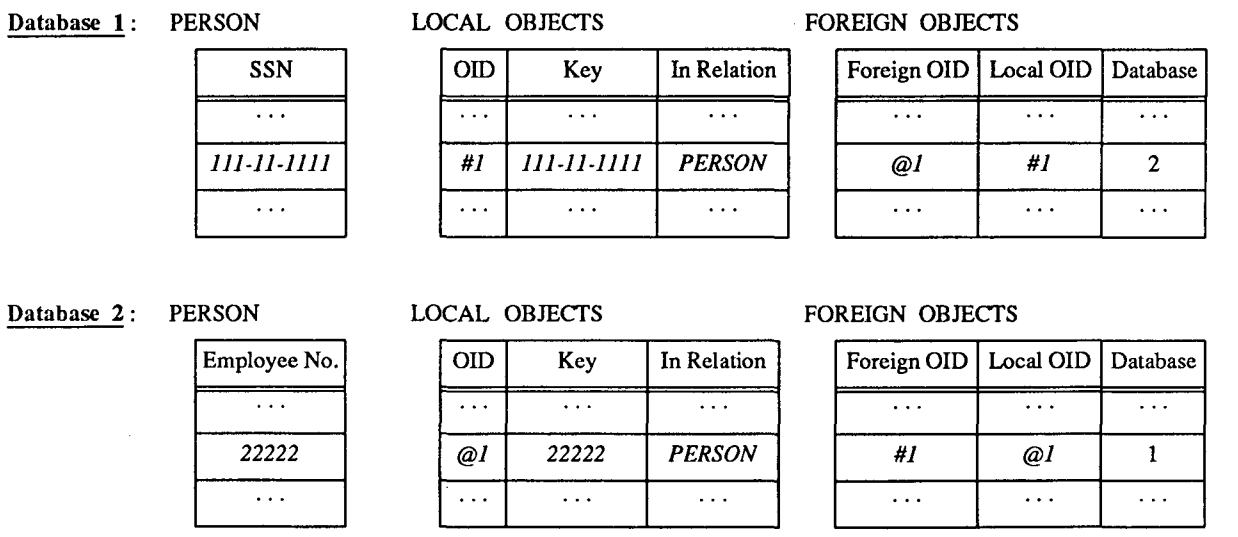


Figure 5.2 Examples of Local-Objects and Foreign-Objects Relations.

r where every key value is associated with the name of the relation in which it appears. Usually updates of keys cause discontinuities in the identification of objects [KHOS86], and require updates of other tuples in order to preserve the referential integrity constraints, that is, have side-effects. In a multidatabase environment these side-effects are amplified if the keys are referenced from other databases. These problems can be avoided by employing *surrogates* as object identifiers (OIDs). Thus, in the local-objects relation for a database *r*, every key value is associated with a unique surrogate, so that references to objects represented in *r* from other *(foreign)* databases will use the surrogate rather than the key value (see attribute OID in relation LOCAL OBJECTS of figure 5.2).

2. The purpose of the *foreign-objects* relation associated with a database *r* is twofold: (i) to list the identifiers of (foreign) objects represented in other databases of the multidatabase, and (ii) to allow the specification of identity relationships between local and foreign objects. All OIDs in the foreign-objects relation are surrogates; every foreign OID is associated with the name of the database in which the foreign OID appears in a local-objects relation. A foreign OID can be associated with the OID of a local object, where the local OID is taken from the local-objects relation; this association defines an *identity* relationship between the corresponding local and foreign objects. In the multidatabase shown in figure 5.2, for example, the persons identified in databases 1 and 2 by keys *111-11-1111* and *22222*, respectively, are defined to be identical.

The consistency of the local-objects and foreign-objects relations is ensured by intra and inter-database referential integrity constraints. (for details see [MARK90b]). The data sharing architecture described above preserves the autonomy of the databases involved in multidatabase, that is, have no effect on the original organization and manipulation of data in these databases.

6. CONCLUSION AND RECOMMENDATIONS

The *Comprehensive Epidemiological Data Resource* (CEDR) project is being developed in a heterogeneous environment, as illustrated by the examples given in this report. Consequently, the CEDR project inherently involves some form of database integration.

We have examined in this report two approaches to the problem of heterogeneous database integration. The first approach to database integration implies *physically* replacing all existing databases with a new, central, database. Such a central database can be implemented employing existing commercial database management systems, and does not require the development of additional mechanisms. The main problem of this approach concerns the complexity and high cost of the integration process, which involves in addition to schema integration, integrating (i.e. converting and merging) data and adapting all the applications that are based on the existing databases to the new central database. Since such a process is beyond the objective of CEDR of providing a mechanism for accessing epidemiologic data (see [PROJ90]), we recommend that:

Recommendation 1 : Physical database integration should not be adopted for CEDR.

The second approach to database integration implies only a *virtual* integration so that the existing databases and applications are preserved. The virtual database integration involves only schemas, and can be carried out incrementally. The main problem of this approach concerns the complexity of the mechanisms needed to support the interoperability of the heterogeneous databases. Since many of these mechanisms are not available, and because developing all these mechanisms would take probably a great deal of time and effort, we recommend that:

Recommendation 2 : Virtual database integration should not be adopted for CEDR.

Following an analysis of the characteristics of the CEDR application, we recommend that:

Recommendation 3 : A compromise between the physical and virtual database integration approaches should be adopted for CEDR.

This approach follows the virtual database integration approach by preserving the existing databases and applications, but compensates the lack of interoperability mechanisms by physically integrating the data subsets provided by the existing databases. The recommended database integration approach is underlined by the architecture shown in figure 6.1:

- every (local) database provides a subset of its data for *export*; the structure (schema) of the export data is specified using a common data model;
- the export data provided by the various (local) databases are integrated into the CEDR database; the

integration process generates a *metadatabase* containing information describing both the CEDR database (e.g. meaning of attributes, domains), and the CEDR data sources (e.g. data quality information) as discussed in [SHOS90].

- a *view definition* capability allows users to view the CEDR data according to their preferences.

Implementing the architecture outlined above involves answering the following questions:

(i) Selecting a Common Data Model. The common data model selected for describing the export data must have enough expressive power for supporting accurate data descriptions. The experience with the IARC data shows that even for small data sets (i.e. for a small number of variables) flat-file descriptions allow the specification of incomplete, and sometimes confusing, data. For example, the

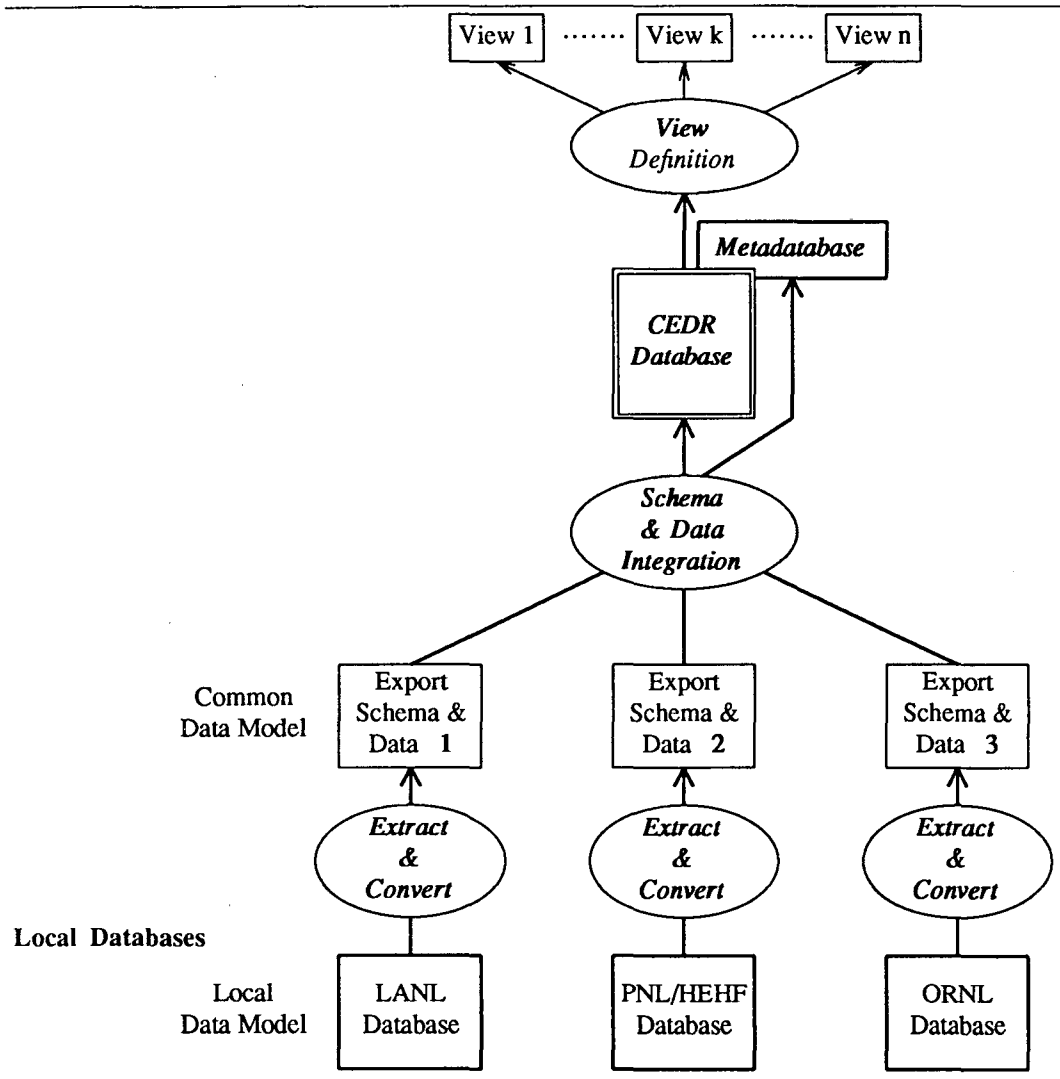


Figure 6.1 Recommended Architecture for *CEDR*.

meaning of *nulls*, which appear frequently in the IARC files, is not always clear (e.g. does a *null* mean that the value is unknown, or not recorded at all, or something else).

(ii) Export Schema Specification. The specification of the structure (schema) of the export data can be carried out either (a) by each local database, (b) as part of the integration process, or (c) by a central authority, such as a committee. The last alternative involves a guidance that can range from specifying only the required data elements (variables) to *standardization*, that is, an exhaustive definition, including details such as formats, names, and units (the IARC protocol is an example of such a standardization effort). The extent of the guidance has important implications on the issues of conversion and evolution, discussed below.

(iii) Conversion Procedures. Every local database must develop procedures for converting the data according to the structure defined for the export data. The complexity of this process depends on the constraints imposed by the export schema specification guidelines mentioned above. Thus, exhaustive schema definitions will imply complex conversion procedures, but will also simplify the subsequent integration process. Conversely, a loose schema specification will imply simple conversion procedures, but will also imply a complex integration process. Conversion procedures must be available in the CEDR metadatabase for verification purposes.

(iv) Database Evolution. The CEDR database is likely to evolve in time. For example, future requirements are likely to lead to the extension of the export data provide by the local databases, such as following the addition of new variables. The integration process must have the capability of coping with such changes.

(v) Data Consistency. Every data element in the CEDR database is a replication of some data in one or several local databases. The consistency of such replicated data must be ensured with special mechanisms. For example, the CEDR metadatabase must be able to support backtracking of any data element from the CEDR database to its source in the local databases.

We outline below two strategies for the implementation of the database integration approach recommended above, and discuss the reasons for preferring one of these strategies over the other.

Strategy 1 : *Standard-driven: conversion at local sites, integration at the CEDR site.*

This strategy involves the following steps:

1. A standard for the integrated CEDR database is developed, usually by a committee. This standard includes the desired data elements (variables), standard formats for the data, and standard units.

2. Each site is asked to extract the data relevant to the standard, and to convert the data according to the standard formats and units.

3. The results are sent from the sites to the CEDR site for integration, which is limited only to *merging* (only the 4th stage of the integration process discussed in section 4.1). The CEDR site verifies that the elements selected and the conversion conform to the standard. If the verification fails, the sites involved are asked to redo the conversion.

Strategy 2 : Standard-independent: conversion and integration at the CEDR site.

1. The sites are asked to send to the CEDR site all datasets (e.g. analysis files, raw data) that may be of interest to the CEDR user community, regardless of any specific standard (at first these datasets may include only analysis files, according to their anticipated value to the users).

2. Each dataset, that is, its data, structure and integrity constraints, is described using an agreed upon exchange format. The CEDR site verifies that the datasets conform to their descriptions.

3. The datasets are described and converted using the common data model at the CEDR site, that is, export schemas are specified for these datasets (this is the *preintegration* stage of integration mentioned in section 4.1). The CEDR site performs the remaining stages (not limited to *merging*) of schema and data integration based on the export schemas and on the data in the datasets sent by the sites.

Note that the current strategy of requiring the sites to generate files corresponding to the IARC standard, conforms to strategy 1. The strategies described above differ in several ways:

(i) Strategy 1 implies that a standard must exist before any data can be sent to the CEDR site for integration. In addition to IARC, the current efforts of determining core variables illustrate the process of developing a standard. Such a process usually takes a long time because of the involvement of committees. By contrast, strategy 2 promotes sending potentially useful data to the CEDR site independent of any standardization effort.

(ii) Strategy 1 limits the data in the CEDR database to that specified by the standard. IARC tried to remedy this restriction by allowing the definition of a file consisting of anything deemed useful by the sending site. The result is a mix of standardized and non-standardized data, without any guidance on how the non-standardized data should be selected and merged. Strategy 2 proposes sending entire datasets (e.g. analysis files), including data that may not be included in a standard. Such data may be useful to an analyst regardless of an accepted standard.

(iii) Strategy 1 depends on the sending sites to carry out the data conversions according to a standard. Strategy 2 shifts the responsibility for conversion to the CEDR site. The advantage of this second approach is that it eliminates the need to go back to the sites for correcting imprecise conversions. A case in point concerns variables representing dates in the IARC files. For example, some variables were sent in a floating point format from which it was possible to derive year and month, but not day. Because of such inconsistencies, a second request for conversion was made.

(iv) Under strategy 1 the need to change a standard, such as adding a new variable, would involve changing the local site conversion processes to the new standard. On the other hand, a similar change under strategy 2 affects conversions at the CEDR site only, provided that the new variables are available in the CEDR database.

(v) Strategy 1 implies that the integration process at the CEDR site is limited only to *merging* (see section 4.1). However, strategy 1 does not eliminate the need for dealing with all the aspects of integration, because standardization itself embodies most of the integration stages, such as solving conflicts, finding common domains, etc. The integration aspects of standardization should be based on a formal integration methodology, rather than being carried out in an ad-hoc manner. Strategy 2 implies that all integration stages are carried out at the CEDR site.

(vi) Strategy 2 does not eliminate standardization, but rather removes it from being on the critical path between various sites and the CEDR site. However, standardization can still determine the definition of *views* of the CEDR database (see figure 6.1). Thus, under strategy 2 the IARC merged files can be implemented as a view over the CEDR database, rather than requiring a conversion process by the sites.

The discussion above leads us to prefer strategy 2 because of (1) its flexibility in terms of the data content of the CEDR database, (2) its non-reliance on multiple sites to perform conversions, and (3) its independence from standardization processes. This strategy simplifies the task of the sites by allowing them to send entire datasets without conversions, promotes sending datasets to CEDR as soon as they are available, and centralizes the integration process. Accordingly, we recommend that:

Recommendation 4 : Strategy 2 outlined above should be adopted for CEDR.

In conclusion, we recommend to further develop and experiment with the architecture outlined above, under strategy 2 above. This will entail (a) developing a schema integration methodology, (b) investigating data conversion strategies, (c) exploring mechanisms for maintaining the consistency of replicated data in CEDR, and (d) develop conventions for describing the datasets.

ACKNOWLEDGEMENT

I want to thank Arie Shoshani for his contribution to the description of the two strategies in section 6, and for the thorough discussions that helped clarify the objectives pursued by the CEDR database.

REFERENCES

[ANSI75] ANSI/X3/SPARC Study Group on DBMS, "Interim Report", *SIGMOD FDT Bulletin,* **7**, 2, 1975.

[BATI86] C. Batini, M. Lenzerini, and S. Navathe, "A comparative analysis of methodologies for database schema integration",*ACM Computing Surveys* **18**,4 (Dec. 1986), pp. 323-364.

[BREI90] Y. Breitbart, "Multidatabase interoperability", *SIGMOD Record* **19**,3 (Sep. 1990), pp.53-60.

[DEMI89] L. DeMichiel, "Resolving database incompatibility: An approach to performing relational operations over mismatched domains", *IEEE Trans. on Knowledge and Database Engineering,* **1**, 4 (Dec. 1989).

[FANK88] P. Fankhauser, W. Litwin, E.J. Neuhold, and M. Schrefl, "Global view definition and multi-database languages- two approaches to database integration", in *Research into Network and Distributed Applications*, R. Speth (ed), Elsevier Science Publishers B.V., 1988, pp. 1069-1082.

[HEIM85] D. Heimbigner and D. McLeod, "A federated architecture for information management", *ACM Trans. on Office Information Systems***3**, 3 (July 1985), pp. 253-278.

[KHOS86] S.N. Khoshafian and G.P. Copeland, "Object identity", in Proc. of the *ACM Conf. on Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, SIGPLAN Notices, vol. 21, no. 11, pp. 417-423, 1986.

[LARS89] J. Larson, S. Navathe, and R. Elmasri, "A theory of attribute equivalence in databases with application to schema integration", *IEEE Trans. on Software Engineering,* **15**, 4 (April 1989), pp. 449-463.

[LITW86] W. Litwin and A. Abdellatif, "Multidatabase interoperability", *IEEE Computer,* **19**, 12 (Dec. 1986), pp. 351-381.

[LITW87] W. Litwin, A. Abdellatif, B. Nicolas, P. Vigier, and A. Zeroual, "MSQL: A multidatabase language", Technical Report 695, INRIA, June 1987.

[LITW89] W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of multiple autonomous databases", Technical Report 2188, Systems Research Center, University of Maryland, 1989.

[MARK88] V.M. Markowitz and J.A. Makowsky, "Incremental restructuring of relational schemas", Proc. of the *4th Int. Conf. on Data Engineering* , IEEE Computer Society Press, Feb. 1988, pp. 276-284.

[MARK90a] V.M. Markowitz and J.A. Makowsky, "Identifying extended entity-relationship object structures in relational schemas", *IEEE Trans. on Software Engineering*, **16**, 8 (Aug. 1990), pp. 777-790.

[MARK90b] V.M. Markowitz, "An architecture for identifying objects in multidatabases", Technical Report LBL-29518, Lawrence Berkeley Laboratory, Sep. 1990.

[MOTR87] A. Motro, "Superviews: Virtual integration of multiple databases", *IEEE Trans. on Software Engineering*, **SE-13**, 7 (July 1987), pp. 785-798.

[NAVA86a] S. Navathe and L. Kerschberg, "Role of dictionaries in information resource management", *Information & Management*, **10**, 1 (Jan. 1986), pp. 21-46.

[NAVA86b] S. Navathe, R. Elmasri, and J. Larson, "Integrating user views in database design", *IEEE Computer* **19**,1, January 1986, pp. 50-62.

[NAVA89] S. Navathe and al, "A federated architecture for heterogeneous information systems", position paper at the NSF Workshop on Heterogeneous Database Systems, Dec. 1989.

[NISO90] National Information Standards Organization, "Guidelines for thesaurus construction, structure, and use", American National Standards Institute, 1990.

[PLAN90] "Comprehensive Epidemiologic Data Resource Information System Project Management Plan", internal document, Lawrence Berkeley Laboratory, August 1990.

[REPO89] Report of the NSF Workshop on Heterogeneous Database Systems, Dec. 1989.

[SHET88] A.P. Sheth, J.A. Larson, A. Cornelio, and S.B. Navathe, "A tool for integrating conceptual schemas and user views", in Proc. of the *4th Int. Conf. on Data Engineering* , IEEE Computer Society Press, 1988, pp. 176-183

[SHET89] A.P. Sheth and S.K. Gala, "Attribute relationships: An impediment in automating schema integration", position paper at the NSF Workshop on Heterogeneous Database Systems, Dec. 1989.

[SHOS90] A. Shoshani, "Metadata management in CEDR", Technical Report, Lawrence Berkeley Laboratory, October 1990.

[ULLM88] J.D. Ullman, *Principles of Database and Knowledge Base Systems*, vol. I, Computer Science Press, 1988.

APPENDIX. SEMANTIC HETEROGENEITY IN CEDR

The IARC protocol captures the common part of the various CEDR applications. For data elements involved in CEDR (and often in the IARC data), the following differences have been observed:

(a) Formats. Different databases have different formats for the same data element (variable). For example, date is represented in different databases with integer, *YY/MM/DD*, and *MMDDYY* formats (*YY* denotes year, *MM* denotes month and *DD* denotes day).

(b) Granularity. Data elements representing measurements differ in granularity levels, such as dose per month or dose per year.

(c) Units. Different databases use different units for the same data element. For example, radiation levels are specified in various units (e.g. rem, msv), depending usually on the devices used for measurement.

(d) Structures. Similar data elements are structured differently in different databases. For example, while in some databases a date consists of month, day, and year, in other databases a date consists of year only.

(e) Incomplete Information. Missing and incomplete information is represented in relational databases employing special *null* values. While in some databases certain data elements are allowed to have null values, in other databases the analogous data elements are not allowed to have null values. Moreover, the meaning of nulls (e.g. "unknown", "not applicable", "not available") varies among databases.

(f) Codes. Codes are used for various reasons, such as for saving storage space, as acronyms for long values, and as medical codes. Codes are often local to the database, and therefore are non-uniform, even when they refer to the same domain (e.g. the same set of diseases).

(g) Definitions. Similar data elements are defined differently in different databases. For example, the definition of socio-economic-status in different databases is underlied by either job classification or year of education; furthermore, the definition of job classification varies in terms of titles and their meanings.

(h) Object Identification. Objects are not uniformly identified in different databases. In particular, different (pseudo) identifiers are used for workers. Global identifiers for workers have to be established in order to correlate information about workers that have moved from site to site (estimated to be about 10% of the worker population).

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
INFORMATION RESOURCES DEPARTMENT
BERKELEY, CALIFORNIA 94720