

A Comparative Analysis of Methodologies for Database Schema Integration

C. BATINI and M. LENZERINI

Dipartimento di Informatica e Sistemistica, University of Rome, Rome, Italy

S. B. NAVATHE

Database Systems Research and Development Center, Computer and Information Sciences Department, University of Florida, Gainesville, Florida 32601

One of the fundamental principles of the database approach is that a database allows a nonredundant, unified representation of all data managed in an organization. This is achieved only when methodologies are available to support integration across organizational and application boundaries.

Methodologies for database design usually perform the design activity by separately producing several schemas, representing parts of the application, which are subsequently merged. Database schema integration is the activity of integrating the schemas of existing or proposed databases into a global, unified schema.

The aim of the paper is to provide first a unifying framework for the problem of schema integration, then a comparative review of the work done thus far in this area. Such a framework, with the associated analysis of the existing approaches, provides a basis for identifying strengths and weaknesses of individual methodologies, as well as general guidelines for future improvements and extensions.

Categories and Subject Descriptors: H.0 [Information Systems]: General; H.2.1 [Database Management]: Data Models; Schema and Subschema; H.2.5: [Database Management]: Heterogeneous Databases; D.2.1: [Requirements/Specifications]: Methodologies

General Terms: Management

Additional Key Words and Phrases: Conceptual database design, database integration, database schema integration, information systems design, models, view integration

INTRODUCTION

1.1 Schema Integration

Database management systems (DBMSs) have been developed in the past two decades using various data models and architectures. The primary data models used for implementation are the hierarchical, network, and relational data models. More recently, several so-called semantic data

models, significantly more powerful than primary data models in representing the application of interest, have been proposed (e.g., see Smith's abstraction hierarchy model [Smith and Smith 1977], the Semantic Data Model [Hammer and McLeod 1981], the TAXIS data model [Mylopoulos et al. 1980], DAPLEX [Shipman 1980], and recently, the Galileo data model [Albano et al. 1985]).

Authors in alphabetical order.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1987 ACM 0360-0300/86/1200-0323 \$1.50

CONTENTS

INTRODUCTION

- I.1 Schema Integration
- I.2 View Integration in Database Design
- I.3 Database Integration
- I.4 Organizational Context for Integration
- I.5 Structure of the Paper

1. METHODOLOGIES FOR SCHEMA

INTEGRATION

- 1.1 An example
- 1.2 Causes for Schema Diversity
- 1.3 Steps and Goals of the Integration Process
- 1.4 Influence of the Conceptual Model on the Integration Process

2. A COMPARISON OF METHODOLOGIES

- 2.1 Introduction
- 2.2 Applicability of Integration Methodologies
- 2.3 Methodologies Viewed as Black Boxes
- 2.4 Gross Architecture of Methodologies
- 2.5 Preintegration
- 2.6 Comparison of Schemas
- 2.7 Conforming of Schemas
- 2.8 Merging and Restructuring

3. CONCLUSIONS AND FUTURE WORK

- 3.1 General Remarks
- 3.2 Missing Aspects
- 3.3 Future Research Directions

APPENDIX 1. A SUMMARY DESCRIPTION OF METHODOLOGIES

APPENDIX 2. THE ENTITY-RELATIONSHIP MODEL

ACKNOWLEDGMENTS

REFERENCES

traditional “data-processing-using-files” approach is that a database management system makes it possible to define an integrated view of relevant data for all applications. This eliminates duplication, avoids problems of multiple updates, and minimizes inconsistencies across applications. Whereas the above claims of the database approach are highly touted, database textbooks and survey literature to date have paid scant attention to this topic. At the same time, research on the problem of integration has proceeded, and most of the researchers have suggested performing the integration activity as a part of the conceptual design step. In this paper we refer to the integration activity by a generic term, *schema integration*, which is defined as the activity of integrating the schemas of existing or proposed databases into a global, unified schema. Schema integration, as defined here, occurs in two contexts:

- (1) *View integration* (in database design) produces a global conceptual description of a proposed database.
- (2) *Database integration* (in distributed database management) produces the global schema of a collection of databases. This global schema is a virtual view of all databases taken together in a distributed database environment.

Since semantic models allow data to be described in a very abstract and understandable manner, they are currently used in designing the conceptual structure of databases. This conceptual activity is called *conceptual database design*. Its goal is to produce an abstract, global view of the data of the application, called *conceptual schema*.

The introduction of a conceptual step in design methodologies is a fairly recent development. It allows designers and users to cooperate in collecting requirements and provides a high-level specification of the data involved in the application. Furthermore, it simplifies the integration of differing perspectives and expectations that various users have of the application.

One of the basic motivations for using the database approach instead of the

The database technology has progressed to a level where thousands of organizations are using databases for their day-to-day, tactical, and strategic management applications. The distributed database management area is also becoming sufficiently well understood, and we expect to see a large number of organizations changing to distributed databases by integrating their current diverse databases.

The contributions to the state of the art of database design methodologies, and in particular schema integration, have been particularly significant in the last ten years. Our goal is to provide first a framework by which the problem of schema integration can be better understood, and second a comparative review of the work done thus far on this problem. Such a framework with an associated analysis of the prevalent

approaches provides

- (1) a conceptual foundation to the problem of schema integration;
- (2) a basis upon which to identify strengths and weaknesses and the missing features about individual methodologies;
- (3) general guidelines for future improvements and extensions to the present approaches.

In the next section we explain the view integration activity; Section I.3 is devoted to database integration. In Section I.4 we elaborate on the motivation for investigating integration. Finally, in Section I.5 we describe the general structure of the remainder of the paper.

I.2 View Integration in Database Design

The problem of *database design* is one of designing the structure of a database in a given environment of users and applications such that all users' data requirements and all applications' process requirements are "best satisfied." This problem has existed ever since DBMSs came into being.

The DBMSs that store and manipulate a database must have a definition of the database in the form of a schema. This is termed the *intension* of the database. The actual values of data in a database are called *instances* or *occurrences* of data. Sometimes they are termed the extension of a database, or just "the database." Whereas the extension of a database keeps changing with time, the intension of the database is supposed to be time invariant. The *database design* problem aims at designing the intension schema of the database, which includes logical specifications such as groupings of attributes and relationships among these groupings (*logical schema*), as well as physical specifications such as the type of access to records, indexes, ordering, and physical placement (*physical schema*). On the basis of this distinction, the corresponding database design activities are termed *logical schema design* and *physical schema design*. Logical schema design involves the problem of designing the conceptual schema and mapping such a schema into the schema definition lan-

guage (or data definition language) of a specific DBMS. Figure 1 shows the phases of database design and the intermediate schema representations. The phases of database design are

- (1) *Requirements Specification and Analysis*. An analysis of the information requirements of various areas within an organization resulting in a preliminary specification of the information needs of various user groups.
- (2) *Conceptual Design*. Modeling and representation of users' and applications' views of information and possibly a specification of the processing or use of the information. The final result of this activity is a conceptual schema that represents a global, high-level description of the requirements.
- (3) *Implementation Design*. Transforming a conceptual schema into the logical schema of a DBMS. The second and third phases taken together are called logical database design.
- (4) *Physical Schema Design and Optimization*. Mapping the logical schema of a database into an appropriate stored representation in a DBMS, including new physical parameters to optimize the database performance against a set of transactions.

Typically, the application design activity proceeds in parallel with database design. Hence, Figure 1 also shows specifications related to applications as the outputs of the last two phases.

As shown in Figure 1, the activity of view integration can be performed at several points of the database design process. It usually is performed during conceptual design. In that case, its goal is to produce an integrated schema starting from several application views that have been produced independently.¹

¹There is a body of work that regards conceptual design as an activity that considers the application as a whole, thus producing a single schema. This includes Batini et al. [1984], Biller and Neuhold [1982], Brodie [1981], Brodie and Zilles [1981], Ceri [1983], Ceri et al. [1981], Chen [1983], Lum et al. [1970], Olle et al. [1982], Rolland and Richards [1982], and Sakai [1981].

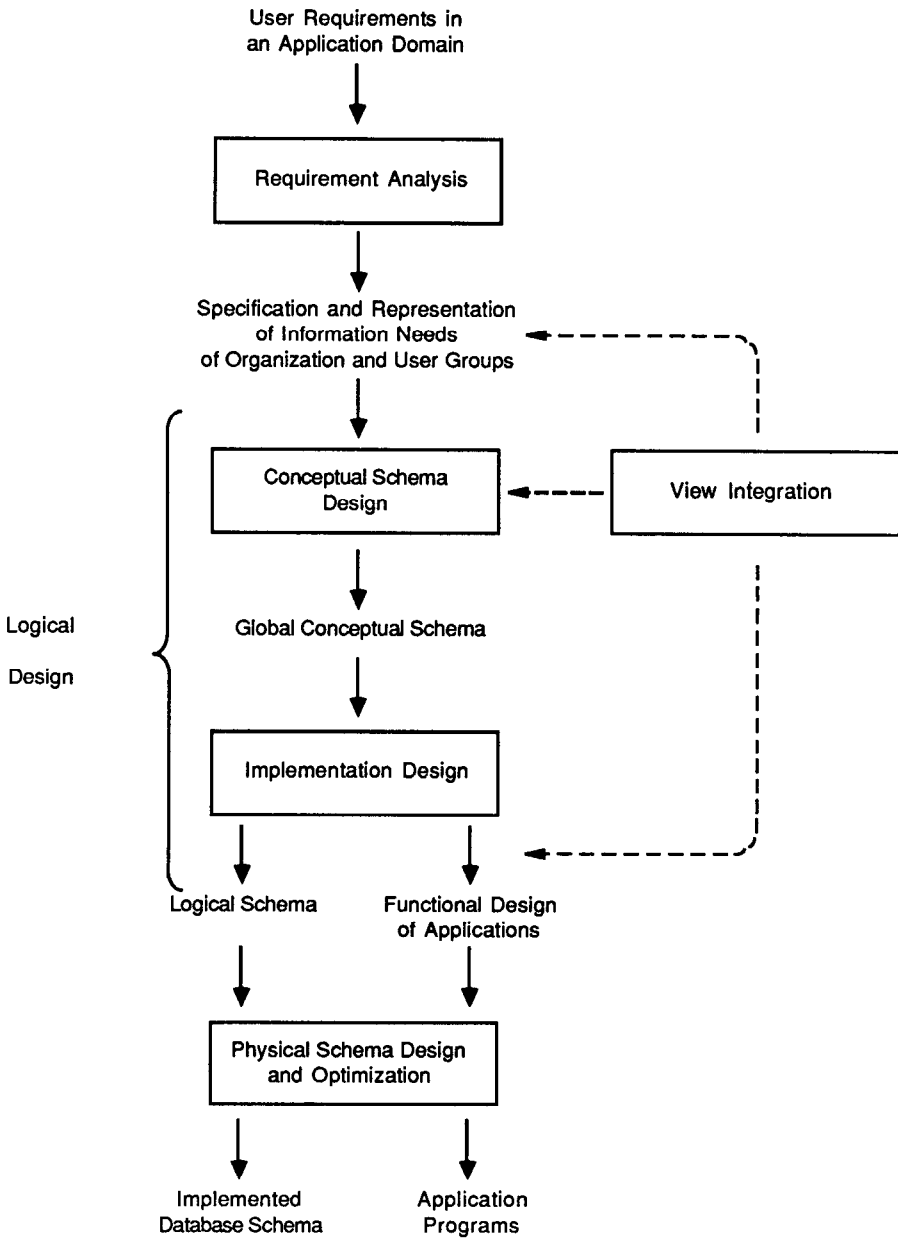


Figure 1. Phases of database design. (Adapted from Navathe and Schkolnick [1978].)

The reason for integration is twofold:

- (1) The structure of the database for large applications (organizations) is too complex to be modeled by a single designer in a single view.
- (2) User groups typically operate independently in organizations and have

their own requirements and expectations of data, which may conflict with other user groups.

Another possibility (Figure 1) is to perform integration even before the “conceptual design” step is undertaken. In this case, view integration still occurs; however,

views are less formal and are mostly in the form of narrative descriptions of requirements. The last possibility shown in the figure is to perform integration after the implementation design step, that is, start from schemas expressed as implementable logical schemas. This is the approach followed in methodologies based strictly on the relational model (see Al-Fedaghi and Scheuermann [1981] and Casanova and Vidal [1983]) that do not advocate a conceptual step and model requirements directly in terms of the relational model.

1.3 Database Integration

Database integration is a relatively recent problem that has appeared in the context of distributed databases. A distributed database is a collection of data that logically belong to the same system but are spread over the sites of a computer network [Ceri and Pelagatti 1984]. Distributed databases and distributed database management systems can be classified into two major categories: *homogeneous*, dealing with local databases having the same data model and identical DBMSs, and *heterogeneous*, having a diversity in data models and DBMSs. The term *Federated Database* is used (e.g., in McLeod and Heimbigner [1980]) to refer to a collection of databases in which the sharing is made more explicit by allowing *export schemas*, which define the sharable part of each local database. Each application is able to design its own global schema by integrating the export schemas.

The above contexts require that an integrated global schema be designed from the local schemas, which refer to existing databases. This too can be considered a database design activity. Existing work on database integration included in our survey implicitly addresses this problem. The authors of these works [Dayal and Hwang 1984; ElMasri et al. 1987; Mannino and Effelsberg 1984a; Motro and Buneman 1981] use a semantic data model as an intermediate model to facilitate the integration. Another implicit assumption they make is that the heterogeneous database management system is able to map the

requests of users—retrievals as well as updates—from such a semantic data model into the actual databases.

The database integration activity is described in a general way in Figure 2. It shows that this activity has as input the local schemas and the local queries and transactions. Most existing work, however, does not explicitly take into account the latter process-oriented information in developing the integrated schema. It is strictly used in mapping the queries (query mapping) between the global and the local levels. Hence, we show the global schema as well as the data and query-mapping specifications to be the outputs of the database integration activity.

1.4 Organizational Context for Integration

Thus far we have pointed out how schema integration arises in database design. As we survey the work on schema integration, it is worthwhile to point out an organizational context for this important area.

There is a growing trend to regard data as an autonomous resource of the organization, independent of the functions currently in use in the organization [National Bureau of Standards 1982]. There is a need to capture the meaning of data for the whole organization in order to manage it effectively. Because of this awareness, integration of data has become an area of growing interest in recent years.

One of the fundamental principles of the database approach is that a database allows a nonredundant, unified representation of all data managed in an organization. This is true only when methodologies are available to support integration across organizational and application boundaries. More and more organizations are becoming aware of the potential of database systems and wish to use them for integrated applications and not just as software for fast retrieval and updating of data.

Even when applications and user groups are structurally disconnected, as in most governmental and large administrative setups, there is something to be gained by having an enterprise-wide view of the data resource. This potentially affords individ-

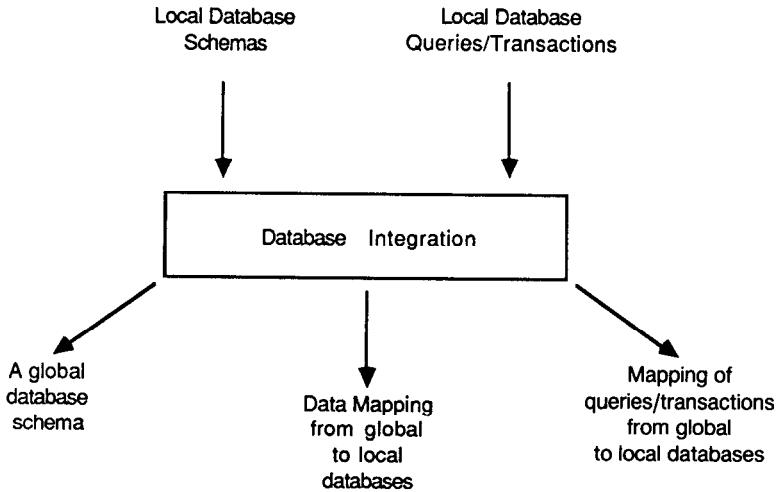


Figure 2. Inputs and outputs of database integration.

ual applications to “build bridges” among themselves and understand how the databases or files relate to one another.

With the increasing use of databases, we expect the integration problem to be more severe and pervasive. New technologies of networking, distributed databases, knowledge-based systems, and office systems will tend to spread the shared use of data in terms of number of users, diversity of applications, and sophistication of concepts. Design, manufacturing, and engineering applications are becoming centered around database management systems. The need for methodologies for integrating data in its diverse conceptual and physical forms is thus expected to increase substantially.

1.5 Structure of the Paper

As described in Section I.1, our main goals are to provide a conceptual foundation for schema integration and perform a detailed comparison of existing work in this area.

It is possible to classify the existing literature into two categories:

- (A) *Complete methodologies* for schema integration. These include view integration and database integration.
- (B) *Related works* addressing specific issues of schema integration.

In the References, the relevant literature is placed under Categories A and B.

In Section 1 we establish the general framework for a comparison of schema integration methodologies. An example introduces the aspects that influence the integration process; we then identify the activities usually performed during schema integration. These activities are used as a basis for comparing methodologies. Finally, we examine the influence of the conceptual model on the overall integration process.

Section 2 is devoted to a detailed comparative analysis of the methodologies. The results of the analysis are presented in the following format:

- (1) Tables illustrating comparative features. The table entries are drawn from the original publications on the methodologies and are not exhaustively explained. However, we extract salient features and trends that are evident in these tables. We highlight the approach of a specific methodology when it explains a specific feature.
- (2) Because of the diversity of the data models (entity–relationship, entity–category–relationship, functional, structural, Navathe–Schkolnick, relational, and generalized entity manipulator) used in the methodologies, we have adopted a uniform treatment of concepts primarily based on the entity–relationship model. The entity–relationship model is briefly summarized in Appendix 2.

In Section 3, we present the conclusions of this investigation, identify missing aspects and open problems, and indicate some research directions.

We compare 12 different complete methodologies. A summary description of each is in Appendix 2. The data model used, inputs and outputs, the general strategy, and special features are briefly described for each methodology.

In order to make the treatment of schema integration uniformly applicable to both the view integration as well as the database integration contexts, we use the following terminology:

General terms used for schema integration: Component Schema, Integrated Schema.

View integration context: user view, conceptual view.

Database integration context: local schema, global schema.

1. METHODOLOGIES FOR SCHEMA INTEGRATION

1.1 An Example

In order to introduce the reader to the main features and problems of schema integration, we present an example. In Figure 3, we show two descriptions of requirements and corresponding possible conceptual schemas that model them.

The following additional information applies to this example:

- (1) The meaning of "Topics" in the first schema is the same as that of "Keyword" in the second schema.
- (2) "Publication" in the second schema is a more abstract concept than "Book" in the first schema. That is, "Publication" includes additional things such as proceedings, journals, monographs, etc.

Figure 4 shows a set of activities that may be performed to integrate the schemas.

Let us look at the two schemas in Figure 4a. Topics and Keywords correspond to the same concept. Since we have to merge the schemas, the names should be unified into a single name. Let us choose the name

Topics. Observe the corresponding change in the second schema as we go from (a) to (b) in Figure 4. When we look at the new schemas (Figure 4b), another difference we notice is that Publisher is present in the two schemas with different types: It is an entity in the first schema and an attribute in the second. The reason for choosing different types (attribute vs. entity) comes from the different relevance that Publisher has in the two schemas. However, we have to conform the two representations if we want to merge them. Therefore we transform the attribute Publisher into an entity in the second schema and add a new attribute, Name, to it (see Figure 4c). We now can superimpose the two schemas, producing the representation in Figure 4d. We have not finished merging yet, since we have to look for properties that relate concepts belonging to different schemas, which were "hidden" previously. This is the case with the subset relationship between the concepts Book and Publication. We can add such a subset relationship to the merged schema, producing the result shown in Figure 4e. Now, to simplify the representation, we can restructure the schema by dropping the properties (relationships and attributes) of Book that are common to Publication. This is allowable since the subset relationship implies that all the properties of publications are implicitly inherited by Book. The final schema is shown in Figure 4f.

1.2 Causes for Schema Diversity

The example of schema integration used above is obviously a "toy example" that highlights some of the basic problems involved. That the integration of realistic sized component schemas can be a complex endeavor is amply evident from this example.

The basic problems to be dealt with during integration come from structural and semantical diversities of schemas to be merged. Our investigation of integration starts with a classification of the various causes for schema diversity, which are different perspectives, equivalence among constructs of the model, and incompatible design specifications.

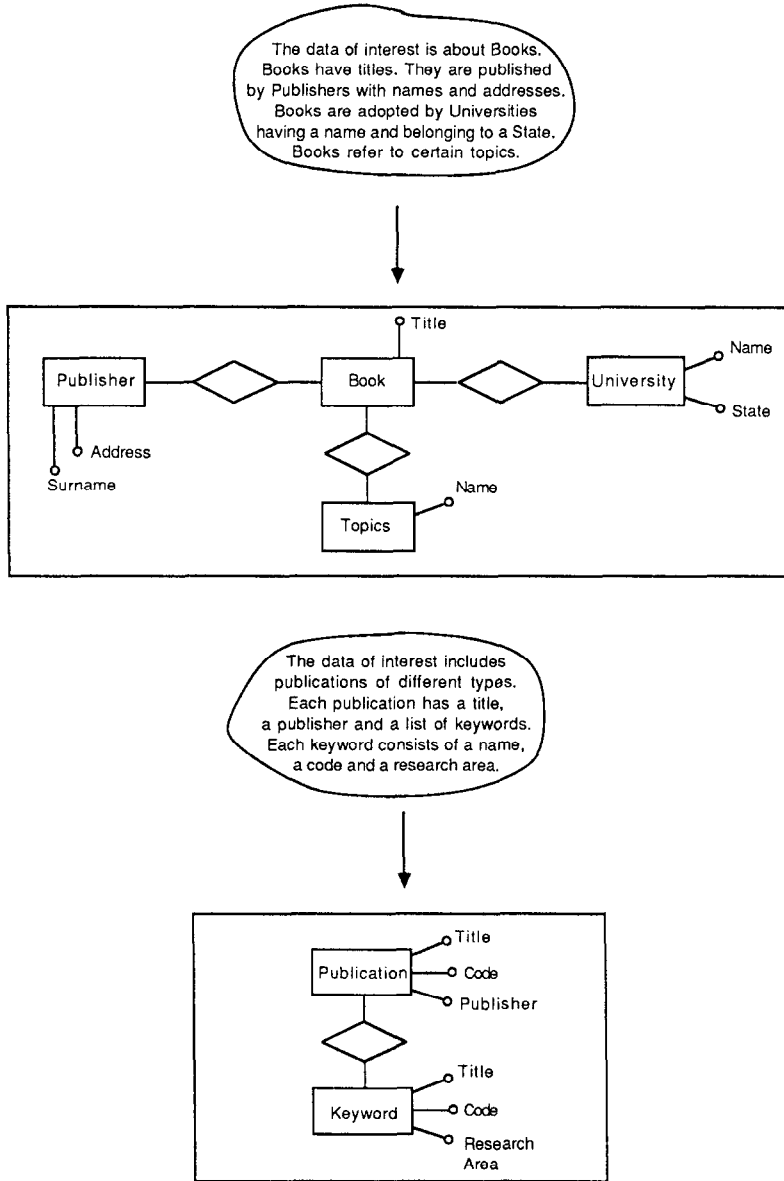


Figure 3. Examples of requirements and corresponding schemas.

1.2.1 Different Perspectives

In the design process, different user groups or designers adopt their own viewpoints in modeling the same objects in the application domain. For instance, in the example in Section 1.1, different names were attached to the same concept in the two views.

Another example is given in Figure 5, in which the two schemas represent information about employees and their departments. In Figure 5a information is modeled by means of the relationship E–D. In Figure 5b, relationship E–P relates the employees with projects, whereas relationship P–D associates projects with departments. It is assumed that an Employee “belongs to”

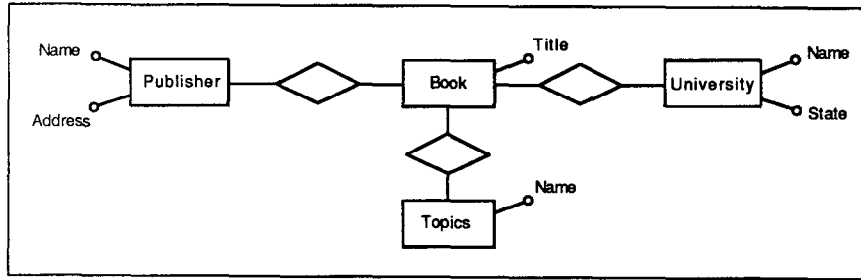


Figure 4a. Original schemas.

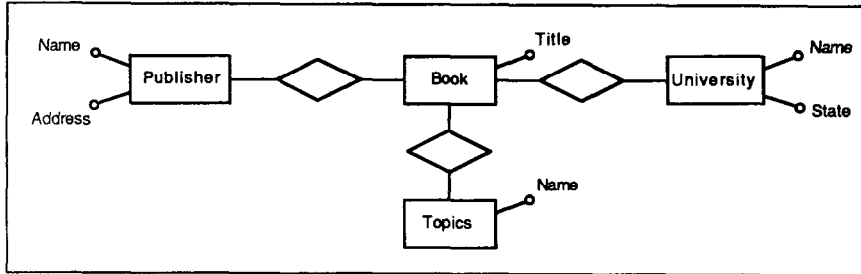
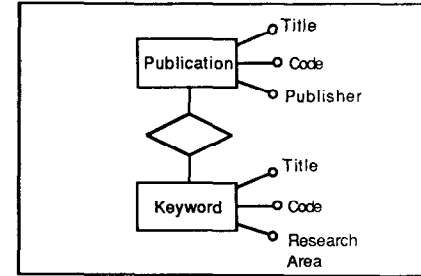


Figure 4b. Choose "Topics" for "Keyword" (Schema 2).

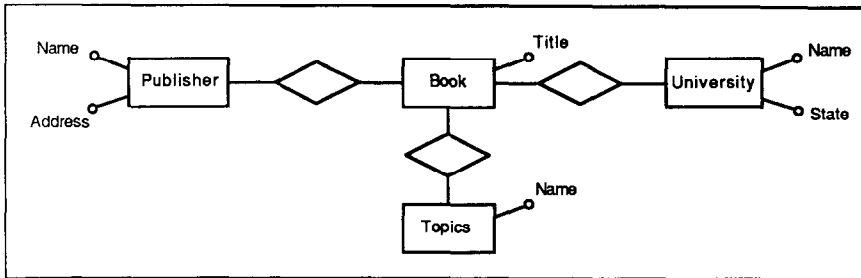
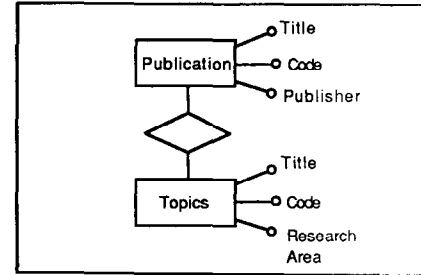


Figure 4c. Make Publisher into an entity (Schema 2).

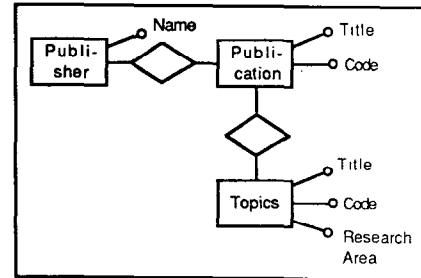


Figure 4d. Superimposition of schemas.

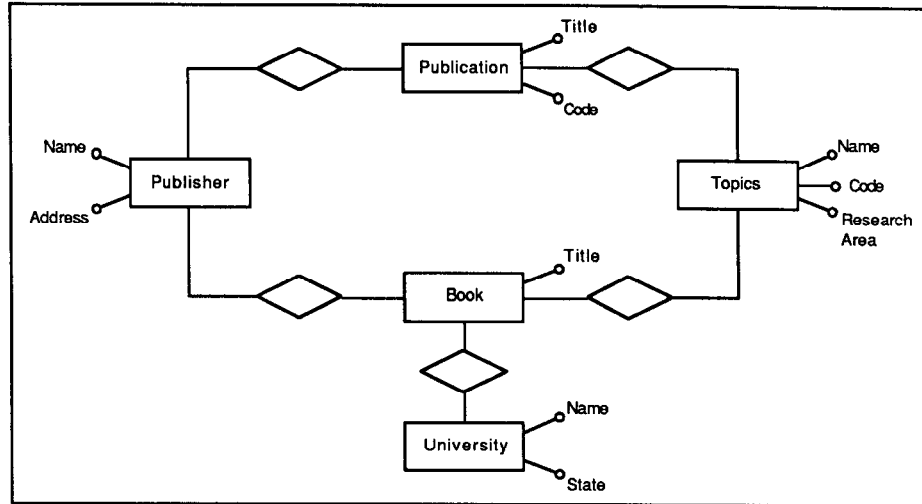


Figure 4e. Creation of a subset relationship.

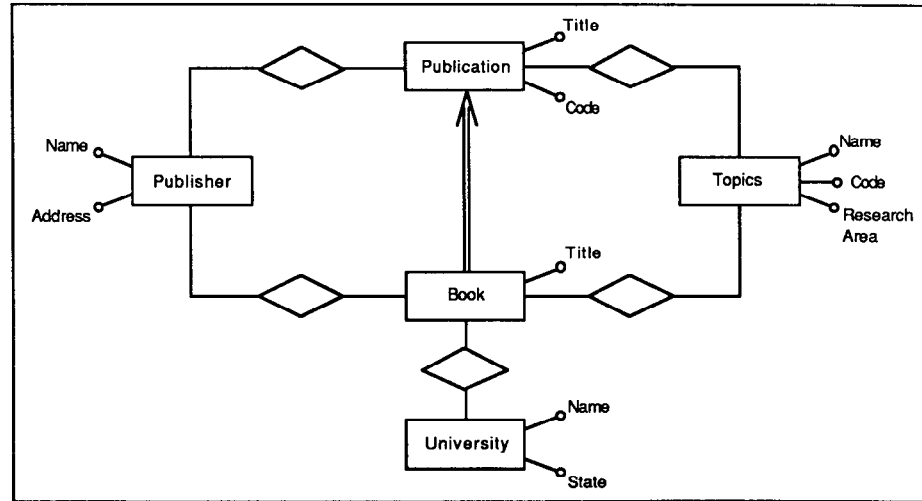


Figure 4f. Drop the properties of Book common to Publication.

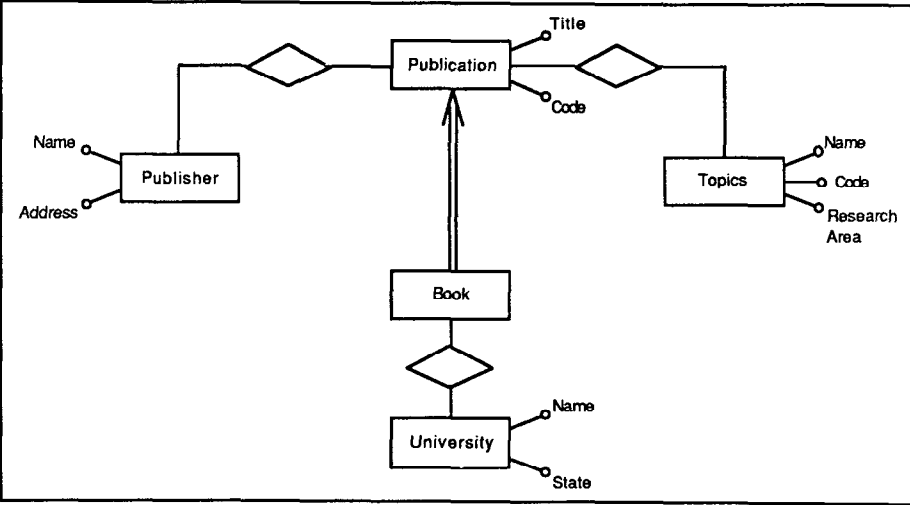


Figure 4. An example of integration.

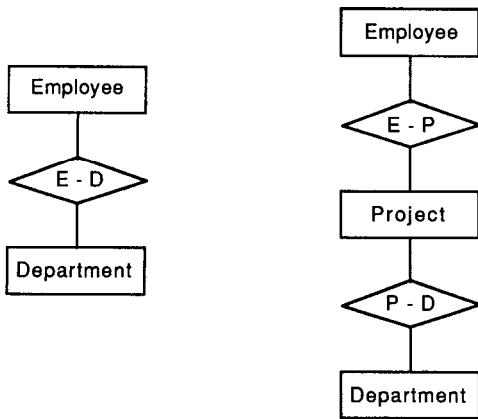


Figure 5. Different perspectives.

those departments that are involved in the projects the employee works on. Therefore the relationship between Employee and Department is perceived as a direct relationship in one schema, whereas it is seen via the entity Project and two relationships in another.

1.2.2 Equivalence among Constructs of the Model

Typically, in conceptual models, several combinations of constructs can model the same application domain equivalently. As a consequence, “richer” models give rise to a larger variety of possibilities to model the same situation. For example, in Figure 3, the association between Book and Publisher was modeled as an attribute of Publisher in one schema and as a relationship between Book and Publisher in the other. Figure 6 shows another example of equivalent constructs. Man and Woman are distinguished by a generalization hierarchy in the first schema, whereas in the second schema they are distinguished by the different values of the attribute Sex.

1.2.3 Incompatible Design Specifications

Erroneous choices regarding names, types, integrity constraints, etc. may result in erroneous inputs to the schema integration process. A good schema integration methodology must lead to the detection of such errors. Schema 1 in Figure 7 erroneously

shows that an Employee is always assigned to a unique project, since the cardinality constraint $1 : n$ has been specified. The correct situation (that an Employee may be assigned to many projects) appears in Schema 2.

These three aspects are concerned with what we can call the common part of the various schemas, that is, the set of concepts of the application domain that are represented in all of the schemas. In other words, the above aspects represent the reasons why the common part may be modeled in different ways in different schemas.

In order to perform integration, it is crucial to single out not only the set of common concepts but also the set of different concepts in different schemas that are mutually related by some semantic properties. We refer to these as *interschema properties*. They are semantic relationships holding between a set of objects in one schema and a different set of objects in another schema. In the rest of this section, we provide a further taxonomy to address correspondences among common concepts and concepts related by interschema properties.

1.2.4 Common Concepts

Owing to the causes for schema diversity described above, it may very well happen that the same concept of the application domain can be represented by different *representations* R_1 and R_2 in different schemas, and several types of *semantic relationships* can exist between such representations. They may be identical, equivalent, compatible, or incompatible:

- (1) *Identical*: R_1 and R_2 are exactly the same. This happens when the same modeling constructs are used, the same perceptions are applied, and no incoherence enters into the specification.
- (2) *Equivalent*: R_1 and R_2 are not exactly the same because different but equivalent modeling constructs have been applied. The perceptions are still the same and coherent. Several definitions of equivalence have been proposed in the literature (see Atzeni et al. [1982], Beeri et al. [1978], Biller [1979], Navathe and Gadgie [1982], Ng et al.

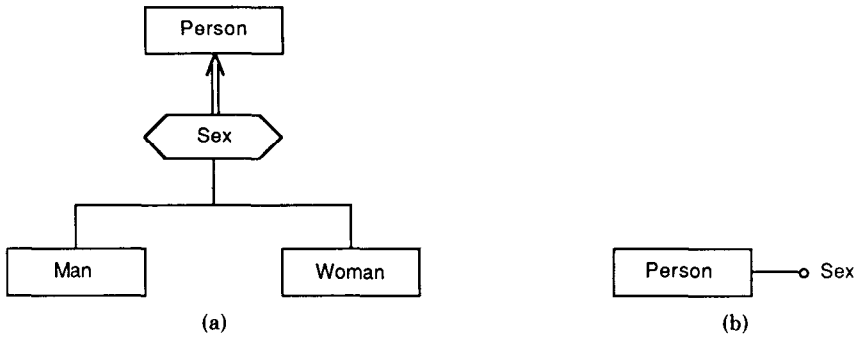


Figure 6. Equivalent constructs. (a) Generalization hierarchy. (b) A single entity.

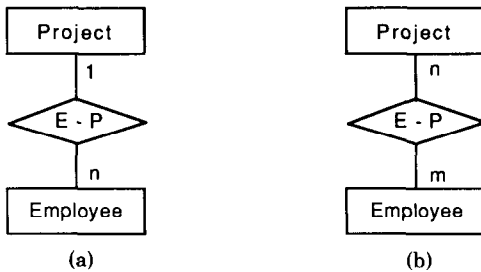


Figure 7. Incompatible design specifications. (a) Schema 1. (b) Schema 2.

[1983], Rissanen [1977]). Although several semantic data models are in existence today, the authors of these models do not provide any criteria for equivalence of concepts. Definitions are typically based on three different types of equivalence:

- (a) *Behavioral*: R_1 is equivalent to R_2 if for every instantiation of R_1 , a corresponding instantiation of R_2 exists that has the same set of answers to any given query and vice versa [Atzeni et al. 1982].
- (b) *Mapping*: R_1 and R_2 are equivalent if their instances can be put in a one-to-one correspondence (e.g., see Rissanen [1977]).
- (c) *Transformational*: R_1 is equivalent to R_2 if R_2 can be obtained from R_1 by applying a set of atomic transformations that by definition preserve equivalence. (Navathe and Gadgie [1982] call this “restructure equivalence.”) This technique is common in other disciplines (e.g., program equivalence).

- (3) *Compatible*: R_1 and R_2 are neither identical nor equivalent. However, the modeling constructs, designer perception, and integrity constraints are not contradictory.
- (4) *Incompatible*: R_1 and R_2 are contradictory because of the incoherence of the specification.

Situations (2), (3), and (4) above can be interpreted as conflicts. Conflicts and their resolutions are central to the problems of integration. A general definition of the term conflict would be as follows:

A *conflict* between two representations R_1 and R_2 of the same concept is every situation that gives rise to the representations R_1 and R_2 not being identical.

1.2.5 Concepts Related by Some Semantic Property

Regarding the concepts in component schemas that are not the same but are related, we need to discover all the *interschema properties* that relate them. In Figure 8, we show two examples of interschema proper-

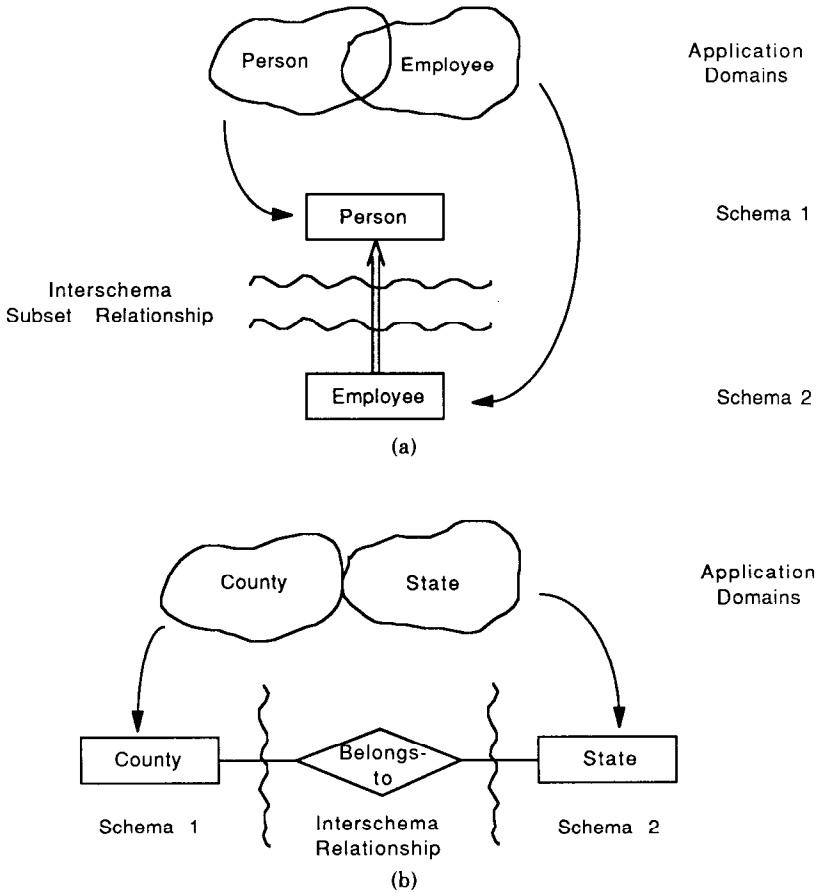


Figure 8. Interschema properties. (a) Example 1. (b) Example 2.

ties. The subset relationship among Person and Employee (Example 1) and the relationship “Belongs-to” between Country and State (Example 2) are interschema properties that could not be perceived in the original component schemas.

1.3 Steps and Goals of the Integration Process

Thus far, we have discussed the nature of the schema integration problem and identified the causes and implications of schema diversity. How do the methodologies accomplish the task of integration? Each methodology follows its own solution procedure. However, any methodology eventually can be considered to be a mixture of the following activities.

1.3.1 Preintegration

An analysis of schemas is carried out before integration to decide upon some integration policy. This governs the choice of schemas to be integrated, the order of integration, and a possible assignment of preferences to entire schemas or portions of schemas. Giving preference to financial applications over production applications is one example of an integration policy that could be set by management.

Global strategies for integration, namely, the amount of designer interaction and the number of schemas to be integrated at one time, are also decided in this phase. Collection of additional information relevant to integration, such as assertions or constraints among views, is also considered to be a part of this phase.

1.3.2 Comparison of the Schemas

Schemas are analyzed and compared to determine the correspondences among concepts and detect possible conflicts. Interschema properties may be discovered while comparing schemas.

1.3.3 Conforming the Schemas

Once conflicts are detected, an effort is made to resolve them so that the merging of various schemas is possible. Automatic conflict resolution is generally not feasible; close interaction with designers and users is required before compromises can be achieved in any real-life integration activity.

1.3.4 Merging and Restructuring

Now the schemas are ready to be superimposed, giving rise to some intermediate integrated schema(s). The intermediate results are analyzed and, if necessary, restructured in order to achieve several desirable qualities. A global conceptual schema may be tested against the following qualitative criteria:

- *Completeness and Correctness.* The integrated schema must contain all concepts present in any component schema correctly. The integrated schema must be a representation of the union of the application domains associated with the schemas.
- *Minimality.* If the same concept is represented in more than one component schema, it must be represented only once in the integrated schema.
- *Understandability.* The integrated schema should be easy to understand for the designer and the end user. This implies that among the several possible representations of results of integration allowed by a data model, the one that is (qualitatively) the most understandable should be chosen.

We make use of the above four phases of schema integration for analyzing and comparing different methodologies.

1.4 Influence of the Conceptual Model on the Integration Process

All of the above issues and activities are strongly influenced by the data model used to express conceptual schemas. The relationship between the comparison and conforming activity and the choice of data model is apparent in all the methodologies that perform these activities "by layers" [Batini et al. 1983; ElMasri et al. 1987; Kahn 1979; Navathe and Gadgil 1982; Teorey and Fry 1982; Wiederhold and ElMasri 1979; Yao et al. 1982]. These layers correspond to the different semantic constructs supported by the model; Table 1 makes an interesting point concerning the specific order of the layers of schema constructs used in the methodologies. The comparison activity focuses on primitive objects first (e.g., entities in the entity-relationship model); then it deals with those modeling constructs that represent associations among primitive objects (e.g., relationships in the entity-relationship model). Note that relational-model-based methodologies do not show up in this table because the relation is their only schema construct.

A few qualitative observations can be made concerning the relative merit of different models.

A simpler data model, that is, one with fewer data-modeling constructs, properties, and constraints has an advantage in conforming and merging activities. This stems from various factors:

- the possibility of type conflicts is smaller;
- the transformation operations are simpler;
- merging involves fewer primitive operations.

On the other hand, a simpler model constitutes a weaker tool in the hands of the designer in discovering similarities, dissimilarities, or incompatibilities. Models with a rich set of type and abstraction mechanisms have the advantage of representing predefined groupings of concepts and allowing comparisons at a higher level of abstraction.

Schema integration comes about when the design of a global schema is attempted

Table 1. Order of Schema Constructs Subjected to Integration

Reference	Phase 1	Phase 2
Batini et al. [1983]	Entities	Relationships
ElMasri et al. [1987]	Object classes	Relationship classes
Kahn [1979]	Entities	Relationships
Navathe and Gadgil [1982]	Objects	Connections
Teorey and Fry [1982]	Aggregations	Generalizations
Yao et al. [1982]	Objects	Functions
Wiederhold and ElMasri [1979]	Primary relations	Connections

using the “divide and conquer” philosophy. It is an inherent attribute or property of this philosophy that the “global characteristics” that cannot be captured by the individual views must be added when a global view becomes available. Consider the relative advantage of the entity–relationship model over the relational model in this respect.

Referring to the example in Section 1.1, adding the subset between Book and Publication allowed us to incorporate a “global characteristic” that was not evident in component schemas. The relational model lacks this modeling feature. Hence, it could only be captured by expressing and enforcing it as a part of the transactions that operate on the global schema or by defining new dependencies such as inclusion interdependencies (e.g., as in Casanova and Vidal [1983]).

The current body of work on the schema integration problem can be divided into two schools: one using the relational or functional models and one using semantic models. Among the semantic models, the entity–relationship model and its variants are dominant.

A few observations can be made when comparing these two schools:

- (1) Methodologies using the relational model ([Al-Fedaghi and Scheuermann 1981; Casanova and Vidal 1983]) make the universal relation schema assumption; that is, every attribute name is unique, across the entire database. As a consequence, problems related to naming and contradictory specifications are ignored. Furthermore, they are not really able to state different perspectives (e.g., contradictory functional dependencies in two views at the start would not be allowed) and natu-

rally avoid dealing with a large subset of the possible conflicts. The semantic-model-based methodologies in general allow a larger amount of freedom in terms of naming, compatible and incompatible design perspectives, etc. Correspondingly, they deal with a much wider spectrum of conflicts.

- (2) The more recent relational-model-based methodologies (e.g., Biskup and Convent [1986], Casanova and Vidal [1983]) use inclusion, exclusion, and union functional dependencies in addition to conventional functional dependencies. An inclusion (exclusion) dependency is used to constrain the set of values associated with a given attribute to be subset of (disjoint from) the set of values corresponding to another attribute. By making use of these dependencies, they claim to achieve the same semantic expressiveness as the semantic models. Owing to the well-defined semantics of the relational model in terms of set theory and dependency theory [Maier 1983; Ullman 1982], they are able to address the problem of minimal redundancy in a formal way.

2. A COMPARISON OF METHODOLOGIES

2.1 Introduction

There are 12 different complete methodologies that we consider (see Category A References). We have placed a summary description of each in Appendix 2, which include the data model used, inputs and outputs, the general strategy, and special features.

In this section, each methodology is analyzed and compared on the basis of some common criteria. In Section 2.2, we concen-

Table 2. Placement of Methodologies

Phases of database design	References
Between requirement analysis and conceptual design	Kahn [1979]
Conceptual design	Batini and Lenzerini [1984], ElMasri et al. [1987], Navathe and Gadgil [1982], Teorey and Fry [1982], Wiederhold and ElMasri [1979]
Implementation design	Al-Fedaghi and Scheuermann [1981], Casanova and Vidal [1983], Yao et al. [1982]

trate on the phases of database design, where the integration methodologies are most applicable. It is seen there that the different methodologies apply to different portions of the design process from requirements analysis to implementation design. We deepen the framework provided in Section 1 by first considering these methodologies as “black boxes” and examine their inputs and outputs. Then we deal with their general structure by examining the procedures that they follow in terms of the four main activities: preintegration, comparison, conforming, merging, and restructuring. Finally, we describe each of these activities in detail.

2.2 Applicability of Integration Methodologies

A majority of the methodologies being analyzed here fall into the class of view integration methodologies. In fact, all except those of Dayal and Hwang [1984], Motro and Buneman [1981], and Mannino and Effelsberg [1984a] belong to this class. That of ElMasri et al. [1987] belongs to both view integration and database integration.

There is little choice in terms of deciding when schemas are integrated in the case of database integration; it has to be performed on the basis of existing local database schemas when a global interface is desired to access them. In contrast, view integration can occur at different times (see Figure 1). It is therefore worthwhile to consider the correspondence between the phases of database design and the various view integration methodologies.

Table 2 shows the phases at which the various view integration methodologies can be considered to be best applicable.

Performing integration during the requirements analysis phase is difficult because user requirements are generally very poorly structured, and are difficult to deal with in terms of a formal methodology involving a semantic analysis. Among the methodologies, only that of Kahn [1979] can be considered applicable to the requirements analysis phase. There, a loosely structured data model is used that resembles those used for collecting requirements specifications.

On the other hand, performing integration during the implementation design phase is difficult because representations at that point do not allow one to make effective use of abstractions. Methodologies such as those of Al-Fedaghi and Scheuermann [1981] and Yao et al. [1982] are able to do integration as a part of the logical design phase by working with the relational (or a functional) data model and various types of dependencies. Pure relational synthesis algorithms (e.g., Bernstein [1976] and Biskup et al. [1979]) can also be considered examples of this approach. As such, they do not deal with the more powerful semantic constructs or abstractions.

The above observations suggest that the preferred phase for integration is the conceptual design phase, where the use of abstraction is very helpful in comparing and conforming different perceptions of the application domain by different user groups.

Another viewpoint regarding the phase when schema integration should be performed may be stated in terms of the following statements:

- (1) Perform integration as early as possible because the cost of carrying erroneous/inconsistent data increases

during the life cycle of the database and the application.

- (2) Perform integration only after complete, correct, minimal, unambiguous representations are available.

This again leads one to the conclusion that schema integration should be placed after requirements analysis but before implementation design. Methodologies [Batini and Lenzerini 1984; ElMasri et al. 1987; Navathe and Gadgil 1982; Teorey and Fry 1982; Wiederhold and ElMasri 1979] indeed confirm this position. We have placed these methodologies under "conceptual design" in Table 2 according to the present terminology. Database integration can be considered to apply more to the conceptual design phase rather than the other two. The above point of view is confirmed by [Dayal and Hwang 1984; ElMasri et al. 1987; Mannino and Effelsberg 1984a; Motro and Buneman 1981] in that for doing database integration they advocate translating heterogeneous logical schemas into conceptual data representations. Hence, all methodologies for database integration [Dayal and Hwang 1984; ElMasri et al. 1987; Mannino and Effelsberg 1984a; Motro and Buneman 1981] are placed in that category.

2.3 Methodologies Viewed as Black Boxes

The basic input to schema integration is a number of component schemas and the basic output is an integrated schema.

Table 3 shows the specific inputs and outputs taken into account by different methodologies. Since Navathe and Gadgil [1982] represented the view integration process with the most comprehensive listing of inputs and outputs, which roughly represent a union of all methodologies, we discuss their terminology:

- *Enterprise View*. Pertinent only to view integration, and not to database integration, this view is an initial conceptual schema which is the enterprise's view of the most important and stable concepts in the application domain. Having such a view at one's disposal makes the activities of comparing and conforming views easier.

- *Assertions*. These correspond to constraints. Intraview assertions are constraints defined on concepts within one schema, whereas interview assertions are constraints among concepts belonging to different views. Methodologies that assume interview assertions to be input implicitly require that some global knowledge pertaining to the diverse applications is supplied to the designer. Modified assertions in the output are revised constraints.
- *Processing Requirements*. These refer to the operations defined on component views. They may be specified in the form of a high-level data manipulation or query language.
- *Mapping Rules*. These define the mapping from queries (operations) applicable to component schemas to queries (operations) against the integrated schema.
- *Statement of Conflicts*. This is a set of conflicts that the designer is not able to resolve and is beyond the scope of the methodology to resolve automatically.

One issue deserving special attention is the treatment of processing requirements. Some methodologies [Al-Fedaghi and Scheuermann 1981; Batini and Lenzerini 1984; Casanova and Vidal 1983; Kahn 1979; Teorey and Fry 1982; Wiederhold and ElMasri 1979] ignore processing requirements totally. Navathe and Gadgil [1982] and Yao et al. [1982] refer to the transactions and queries on component schemas that have to be supported after integration. Navathe and Gadgil [1982] show that this support of processing requirements is provided by a set of mapping rules. In Dayal and Hwang [1984] and Motro and Buneman [1981] the query modification process is addressed in detail to deal with the processing of local queries on the global database. Batini et al. [1983] and Yao et al. [1982] consider the problem of query modification during view integration.

We can conclude that a complete treatment of processing requirements during integration is not present in any of the methodologies surveyed. Some recent proposals have been made to combine process design with database design [Carswell and Navathe 1986].

Table 3. Inputs and Outputs

Reference	Inputs	Outputs
Al-Fedaghi and Scheuermann [1981]	n External views	n External schemas Conceptual schema Mapping between external schemas and conceptual schema
Batini and Lenzerini [1984]	User schemas Weights for schemas Enterprise schema	Global schema
Casanova and Vidal [1983]	User views	Conceptual schema
Dayal and Hwang [1984]	Local schemas of existing databases Queries	Global interface to databases Modified queries
ElMasri et al. [1987]	Local schemas Interschema assertions	Global schema Mapping rules
Kahn [1979]	Local information structures	Global information structure
Motro and Buneman [1981]	Logical schemas Database queries	Superview Modified queries
Mannino and Effelsberg [1984a]	Local schemas Interschema assertions about entities and attributes	Global schema Mapping rules Definition of integration schema objects
Navathe and Gadgil [1982]	Enterprise view Local views Interview assertions Intraview assertions Processing requirements	Global view Mapping rules Modified assertions Conflicts
Teorey and Fry [1982]	Information, application, event, corporate perspectives Policy guidance and rules	Global information structure Conflicts
Yao et al. [1982]	Views Processing specifications	Global view Modified processing specification
Wiederhold and ElMasri [1979]	Two schemas	Global schema

The form in which the inputs and outputs exist in an integration system (which may be partly automated) is not stated explicitly by any of the authors considered. It is obvious that in order to process the schemas in an automated environment, they must be expressed in some well-defined language or some internal representation using data structures.

2.4 Gross Architecture of Methodologies

Let us consider the four activities of the integration process. In Table 4, we show the steps that are performed by each of the methodologies and the looping structure present in them.

It is possible to classify the methodologies into four groups on the basis of

Table 4:

- (1) Those that perform a repetitive comparison, conforming, and merging of schemas, and avoid the need to restructure later [Mannino and Effelsberg 1984a; Navathe and Gadgil 1982; Wiederhold and ElMasri 1979].
- (2) Those that perform most of the activities during and after the merging of schemas. They include Steps 3 and 4 only and avoid comparison and conforming of the schemas [Al-Fedaghi and Scheuermann 1981; Casanova and Vidal 1983; Motro and Buneman 1981; Teorey and Fry 1982; Yao et al. 1982].
- (3) Those that perform all four activities [Batini and Lenzerini 1984; Dayal and

Table 4. Schema Integration Activities

References	Preintegration (Step 1)	Compare (Step 2)	Conform (Step 3)	Merging (Step 4a)	Restructuring (Step 4b)
Al-Fedaghi and Scheuermann [1981]	—	—	—	X \longleftrightarrow X	
Batini and Lenzerini [1984]	—	X \longleftrightarrow X	—	X \longrightarrow X	X \longrightarrow X
Casanova and Vidal [1983]	—	—	—	X \longrightarrow X	X \longrightarrow X
Dayal and Hwang [1984]	—	X \longrightarrow X	—	X \longrightarrow X	X \longleftrightarrow X
ElMasri et al. [1987]	X \longrightarrow X	X \longleftrightarrow X	—	X \longrightarrow X	X \longrightarrow X
Kahn [1979]	—	X \longrightarrow X	—	X \longrightarrow X	X \longrightarrow X
Motro and Buneman [1981]	—	—	—	X \longleftrightarrow X	
Mannino and Effelsberg [1984a]	X \longrightarrow X	X \longleftrightarrow X	—	X \longrightarrow X	—
Navathe and Gadgil [1982]	X \longrightarrow X	X \longleftrightarrow X	—	X \longrightarrow X	—
Teorey and Fry [1982]	—	—	—	X \longrightarrow X	X \longrightarrow X
Yao et al. [1982]	—	—	—	X \longleftrightarrow X	
Wiederhold and ElMasri [1979]	—	X \longleftrightarrow X	—	X \longrightarrow X	—

Hwang 1984; ElMasri et al. 1987; Kahn 1979].

- (4) Those that explicitly mention preintegration analysis [ElMasri et al. 1987; Mannino and Effelsberg 1984a; Navathe and Gadgil 1982].

On the basis of the looping structure alone, the following similarities can be observed:

- (1) Casanova and Vidal [1983] and Teorey and Fry [1982] have a “no-feedback” approach to integration. They only perform the merging and restructuring steps.
- (2) Al-Fedaghi and Scheuermann [1981]; Dayal and Hwang [1984], Motro and Buneman [1981], and Yao et al. [1982] are similar to the above group in that they perform only merging and restructuring; however, they allow a feedback between these two steps.
- (3) Kahn [1979], Mannino and Effelsberg [1984a], Navathe and Gadgil [1982], Wiederhold and ElMasri [1979] provide a global loop from the end of the process to the initial comparison activity. Kahn [1979] includes the restructuring step, whereas the others do not.
- (4) Finally, Batini and Lenzerini [1984] and ElMasri et al. [1987] cover all the steps; moreover, they provide an iterative execution of comparison and conforming steps before any merging is

attempted. As such, they appear to have the maximum interaction with the user/designer.

2.5 Preintegration

As shown in Table 4, only three methodologies [ElMasri et al. 1987; Mannino and Effelsberg 1984a; Navathe and Gadgil 1982] explicitly mention preintegration. They basically propose a collection of correspondences among schemas in the form of constraints and assertions among component schemas. These specifications are used, for example, to relate names, to establish that an object in one schema is the result of some operation on a set of objects in another schema, etc.

For all methodologies, whether or not preintegration is explicitly mentioned, the sequencing and grouping of schemas for integration has to be considered. In this section we describe the different strategies that address this problem.

The first step, choice of schemas, involves processing component schemas in some sequence. In general, the number of schemas considered for integration of each step can be $n \geq 2$. Figure 9 shows four possible variations termed *integration-processing strategies*. Each strategy is shown in the form of a tree. The leaf nodes of the tree correspond to the component

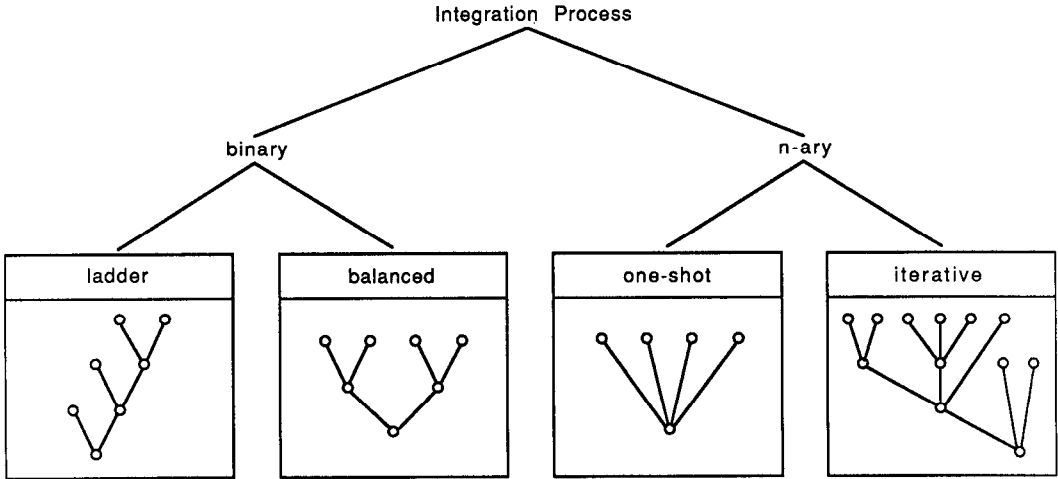


Figure 9. Types of integration-processing strategies.

Table 5. Integration-Processing Strategies

Reference	Type of integration-processing strategy	Balance of strategy
Al-Fedaghi and Scheuermann [1981]	One-shot <i>n</i> -ary	—
Batini and Lenzerini [1984]	Binary	Ladder
Casanova and Vidal [1983]	Binary	Balanced
Dayal and Hwang [1984]	Binary	No claim
ElMasri et al. [1987]	One-shot <i>n</i> -ary	—
Kahn [1979]	Binary	No claim
Motro and Buneman [1981]	Binary	No claim
Mannino and Effelsberg [1984a]	Binary among families	No claim
Navathe and Gadgil [1982]	Iterative <i>n</i> -ary	—
Teorey and Fry [1982]	Binary	Balanced
Yao et al. [1982]	One-shot <i>n</i> -ary	—
Wiederhold and ElMasri [1979]	Binary	Ladder

schemas, and the nonleaf nodes correspond to intermediate results of integration. The root node is the final result. The primary classification of strategies is binary versus *n*-ary.

Binary strategies allow the integration of two schemas at a time. They are called *ladder strategies* when a new component schema is integrated with an existing intermediate result at each step. A binary strategy is *balanced* when the schemas are divided into pairs at the start and are integrated in a symmetric fashion (see Figure 9, balanced).

N-ary strategies allow integration of *n* schemas at a time ($n > 2$). An *n*-ary strategy is *one shot* when the *n* schemas are integrated in a single step; it is *iterative* otherwise. The latter is the most general case.

Table 5 is a comparison of methodologies along two dimensions: binary versus *n*-ary and the nature of balancing.

We now comment on the specific features pertaining to the above classes of strategies.

The advantage of binary strategies is in terms of simplifying the activities of comparison and conforming at each integration step. It is evident from the table that most of the methodologies agree in adopting a binary strategy because of the increasing complexity of the integration step with respect to the number of schemas to be integrated. In general, the merging algorithm for *n* schemas can be shown to be n^2 in complexity. Hence, keeping *n* down is desirable from the standpoint of complexity. The disadvantages of binary strategies are an increased number of integration opera-

tions and the need for a final analysis to add missing global properties.

The motivation behind the ladder processing strategy comes from two reasons:

- (1) Component schemas can be picked up for integration in the decreasing order of their relevance (or “weights,” as Batini and Lenzerini [1984] call them);
- (2) There is an inherent importance associated with an already existing partially integrated schema.

An integration step could take advantage of this situation by resolving conflicts in favor of the partially integrated schema. For instance, an enterprise view (see Section 2.3) is frequently viable in an organization. Choosing it as the initial schema makes the detection and resolution of dissimilarities more efficient.

A binary balanced strategy has been proposed only by Teorey and Fry [1982]. They justify it on the basis of minimizing the number of comparisons among concepts in the schemas.

The work of ElMasri and Navathe [ElMasri 1980; Navathe et al. 1984] are good examples of one-shot n -ary strategies. They consider that during Step 2, an analysis of the n schemas is performed together. After collecting, analyzing, and validating all the interview assertions, they perform the integration in a single step. The obvious advantages of n -ary integration are:

- (1) A considerable amount of semantic analysis can be performed before merging, avoiding the necessity of a further analysis and transformation of the integrated schema;
- (2) The number of steps for integration is minimized.

The recommended procedure given by Navathe and Gadgil [1982] is an iterative n -ary strategy where “equivalence groups” of user views are initially formed; the views within the groups are merged first, creating intermediate integrated schemas that are again analyzed and grouped iteratively.

Not all the analyzed methodologies state what strategy they adopt. Hardly any (except Teorey and Fry [1982]) make any statement about balancing.

2.6 Comparison of Schemas

The fundamental activity in this step consists of checking all conflicts in the representation of the same objects in different schemas. Methodologies broadly distinguish two types of conflicts (see Table 6): naming conflicts and structural conflicts. We now examine each in detail.

2.6.1 Naming Conflicts

Schemas in data models incorporate names for the various objects represented. People from different application areas of the same organization refer to the same data using their own terminology and names. This results in a proliferation of names as well as a possible inconsistency among names in the component schemas. The problematic relationships among names are of two types:

- (1) *Homonyms*: When the same name is used for two different concepts, giving rise to inconsistency unless detected. Consider the two schemas shown in Figure 10. Both schemas include an entity named EQUIPMENT. However, the EQUIPMENT in Figure 10a refers to Computers/Copiers/Mimeographic machines, whereas in Figure 10b it refers to pieces of furniture as well as air conditioners. It is obvious that merging the two entities in the integrated schema would result in producing a single entity for two conceptually distinct objects.
- (2) *Synonyms*: When the same concept is described by two or more names. Unless different names improve the understanding of different users, they are not justified.

An example appears in Figure 11, where CLIENT and CUSTOMER are synonyms; the entities with these two names in the two schemas refer to the same real-world concept. In this case, keeping two distinct entities in the integrated schema would result in modeling a single object by means of two different entities.

The motivation behind establishing naming correspondences and discovering

Table 6. Naming and Structural Conflicts

Reference	Naming conflicts	Structural conflicts
Al-Fedaghi and Scheuermann [1981]	—	—
Batini and Lenzerini [1984]	Homonyms Synonyms	Type inconsistencies Integrity constraints conflicts
Casanova and Vidal [1983]	—	—
Dayal and Hwang [1984]	Homonyms Synonyms	Schema level conflicts: Scale differences Structural differences Differences in abstraction Data level inconsistencies: Various levels of obsolescence and reliability
ElMasri et al. [1987]	Homonyms Synonyms Attribute equivalence assertions Entity class equivalence	Open ended treatment of conflicts, specifically: Differences in abstraction levels Differences in roles, degree, and cardinality constraints of relationships
Kahn [1979]	Homonyms Synonyms	Cardinality ratio conflicts
Motro and Buneman [1981]	—	—
Mannino and Effelsberg [1984a]	Use of qualified names Attribute equivalence specification	Differences in abstractions
Navathe and Gadgil [1982]	Homonyms Synonyms	Dependency conflicts Redundancy conflicts Modeling conflicts
Teorey and Fry [1982]	—	—
Yao et al. [1982]	—	—
Wiederhold and ElMasri [1979]	—	Cardinality ratio conflicts

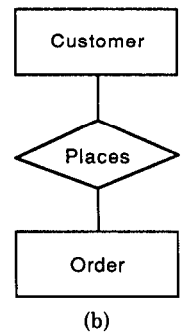
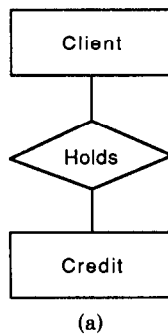
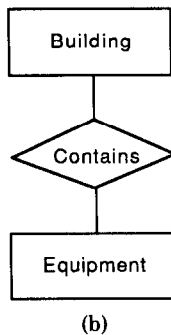
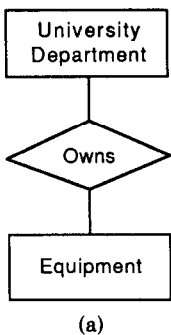


Figure 10. Example of homonyms.

Figure 11. Example of synonyms.

homonyms and synonyms is to be able to determine the four kinds of semantic relationships among component schemas that we introduced in Section 1.2. Note that whereas homonyms can be detected by

comparing concepts with the same name in different schemas, synonyms can only be detected after an external specification.

Data dictionaries have been advocated as a useful adjunct tool to schema integration

methodologies for a better management of names [Navathe and Kerschberg 1986].

Methodologies [Al-Fedaghi and Scheuermann 1981; Casanova and Vidal 1983; Motro and Buneman 1981; Teorey and Fry 1982; Wiederhold and ElMasri 1979; Yao et al. 1982] make no mention of naming correspondences, probably as a result of an implicit assumption that such correspondences are preestablished and thus no naming conflicts can arise (see also the discussion of the relational model in Section 1.4). In ElMasri et al. [1987] a full naming convention

schemaname.objectname

for objects and

schemaname.objectname.attributename

for attributes is adopted to assure uniqueness of names. As a consequence, homonyms cannot arise. The synonym problem still remains and must be dealt with via the establishment of attribute classes. There is also a cross-reference lexicon of names maintained to keep information on synonyms. In Batini and Lenzerini [1984] and ElMasri et al. [1987] the integration system automatically assigns a "degree of similarity" to pairs of objects, based on several matching criteria. Users are presented with the similarity information to help them detect synonyms.

A type of homonyms arises when for the same concept there is a match on names but no match on the corresponding sets of instances. They can occur at various levels of abstraction. For example, at the attribute level, size refers to dress size (a single integer code) in one schema, whereas it refers to trouser size (a pair of integers) in another schema. At the entity level, STUDENT refers to all students in the database kept in the registrar's office, whereas it refers to married students only in the married-student-housing database.

2.6.2 Structural Conflicts

We use the term *structural conflicts* to include conflicts that arise as a result of a different choice of modeling constructs or integrity constraints. They can be traced

back to the three reasons for schema diversity described in Section 1.2. Table 6 lists the different kinds of structural conflicts that are taken into account in various methodologies. Here we present a classification of structural conflicts that is independent from the various terminologies and from the specific characteristics of the different data models adopted in the methodologies. Such a classification distinguishes the following kinds of conflicts:

- (1) *Type Conflicts*. These arise when the same concept is represented by different modeling constructs in different schemas. This is the case when, for example, a class of objects is represented as an entity in one schema and as an attribute in another schema.
- (2) *Dependency Conflicts*. These arise when a group of concepts are related among themselves with different dependencies in different schemas. For example, the relationship Marriage between Man and Woman is 1:1 in one schema, but $m:n$ in another accounting for a marriage history.
- (3) *Key Conflicts*. Different keys are assigned to the same concept in different schemas. For example, SS# and Emp-id may be the keys of Employee in two component schemas.
- (4) *Behavioral Conflicts*. These arise when different insertion/deletion policies are associated with the same class of objects in distinct schemas. For example, in one schema a department may be allowed to exist without employees, whereas in another, deleting the last employee associated with a department leads to the deletion of the department itself. Note that these conflicts may arise only when the data model allows for the representation of behavioral properties of objects.

Another activity typically performed during the schema comparison step is the discovery of interschema properties. Methodologies usually consider this discovery to be a by-product of conflict detection. If any interschema properties are discovered during this step, they are saved and processed

Table 7. Schema Transformations Performed by Methodologies

Reference	Conform	Merge and restructure
Al-Fedaghi and Scheuermann [1981]	—	Removal of redundant dependencies
Batini and Lenzerini [1984]	Type transformations Restructuring Renaming	Subsetting Aggregation Restructuring
Casanova and Vidal [1983]	—	Optimization
Dayal and Hwang [1984]	—	Include Integration by generalization Define supertype Define subtype Scale unifying Renaming
ElMasri et al. [1987]	Modify assertions Renaming	Remove redundant relationships
Kahn [1979]	Renaming	Redundancy elimination
Mannino and Effelsberg [1984a]	Algebraic operations	Create generalization hierarchies Create subtype
Motro and Buneman [1981]	—	Meet Fold Aggregate Join Add Delete
Navathe and Gadgil [1982]	—	Attribute enhancement Attribute creation Restriction
Teorey and Fry [1982]	—	Aggregation Generalization
Yao et al. [1982]	—	Removal of functions
Wiederhold and ElMasri [1979]	—	Subsetting

during schema merging [ElMasri et al. 1987] or schema restructuring [Batini and Lenzerini 1984].

In general, both the discovery of conflicts and the interschema properties are aided by a strong interaction between the designer and the user. This is the position advocated by [Batini and Lenzerini [1984], ElMasri et al. [1987], Kahn [1979], Mannino and Effelsberg [1984a], and Navathe and Gadgil [1982].

2.7 Conforming of Schemas

The goal of this activity is to conform or align schemas to make them compatible for integration. Achieving this goal amounts to resolving the conflicts, which in turn re-

quires that schema transformations be performed. In order to resolve a conflict, the designer must understand the semantic relationships among the concepts involved in the conflict. Sometimes conflicts cannot be resolved because they arose as a result of some basic inconsistency. In this case, the conflicts are reported to the users, who must guide the designer in their resolution.

The concept of schema transformation is central to conflict resolution and therefore to the conforming activity. Since methodologies also perform schema transformations during merging and restructuring, in Table 7 we introduce a comprehensive taxonomy of all types of transformations.

From this table it is clear that a limited number of transformations are proposed

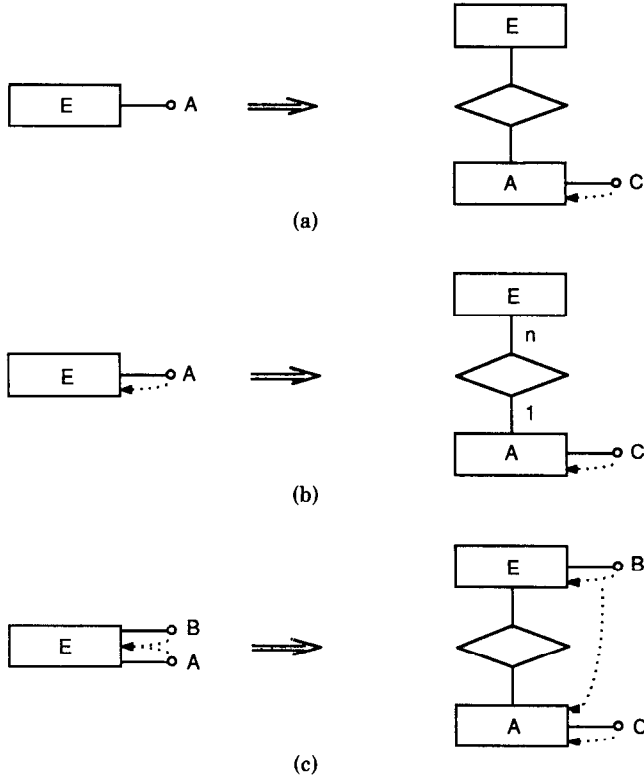


Figure 12. Transformation of an attribute into an entity.

for conflict resolution. Simple renaming operations are used for solving naming conflicts by most methodologies. With regard to other types of conflicts, the methodologies do not spell out formally how the process of resolution is carried out; however, an indication is given in several of them as to how one should proceed. For example, when dealing with equivalence, Batini and Lenzerini [1984] suggest that atomic concepts be transformed (i.e., transform entities/attributes/relationships among one another) to reach a common canonical representation of the schemas.

We show in Figure 12 three examples of transforming an attribute into an entity, as suggested by Batini and Lenzerini [1984]. The dashed lines in these figures specify identifiers. In Figure 12a attribute A is not an identifier. It is shown to be transformed into an entity. In Figure 12b, attribute A, which is an identifier, becomes an entity in the new schema; entity A now provides

identification to entity E (since 1:n means that every instance of A participates only once in the relationship with E). In Figure 12c attribute A is only a part of an identifier and so in the new structure, entity A becomes a part of a compound identifier for entity E.

It is interesting to note that among the methodologies surveyed, none provide an analysis or proof of the completeness of the schema transformation operations from the standpoint of being able to resolve any type of conflict that can arise.

All the methodologies take the goal of the conforming activity to be the construction of a single "consensus schema" by possibly changing some user views. This is consistent with the ANSI/SPARC [Klug and Tsichritzis 1977] three-schema architecture in which the conceptual schema is a unified representation of the whole application, whereas individual perspectives are captured by external schemas.

2.8 Merging and Restructuring

The activities usually performed by methodologies during this phase require different kinds of operations to be performed on either the component schemas or the temporary integrated schema. In order to establish a common framework for this phase, we assume that all methodologies first merge the component schemas by means of a simple superimposition of common concepts, and then perform restructuring operations on the integrated schema obtained by such a merging. Table 8 shows the transformations proposed in the methodologies for this step. Each transformation is performed in order to improve the schema with respect to one of the three qualities described in Section 1.3, namely, completeness, minimality, and understandability. We now analyze each quality separately.

2.8.1 Completeness

To achieve completeness, the designer has to conclude the analysis and addition of interschema properties that is initiated in previous design steps. In Figure 8 we showed examples of interschema properties. In Table 8 we present a comprehensive list of interschema properties mentioned in the methodologies. Note that "subsetting" is the interschema property used by most methodologies. In fact, it is considered to be the basis for accommodating multiple user perspectives on comparable classes of objects.

Batini and Lenzerini [1984], Dayal and Hwang [1984], Mannino and Effelsberg [1984a], Motro and Buneman [1981], Teorey and Fry [1982], and Wiederhold and ElMasri [1979] propose suitable transformations for introducing subset-generalization relationships in the integrated schema (subsetting, integration by generalization, define subtype, etc. are the names of such transformations). In Motro and Buneman [1981], "meet" is the transformation that produces a common generalization of two classes. Such a transformation is based on the existence of a common key for the two classes. On the other hand, "join" produces a common subtype for the two classes. It is used when a class is needed in the inte-

grated schema for the representation of the set of instances that are common to two different classes.

Other types of interschema properties are concerned with aggregation relationships among classes. Batini and Lenzerini [1984], Motro and Buneman [1981] and Teorey and Fry [1982], propose specific transformations for introducing new relationships in the integrated schema so that aggregation among classes coming from different component schemas can be represented.

Finally, there is a set of transformations that introduces new concepts in order to convey all the information represented in the component schemas. In Navathe and Gadgil [1982] "attribute creation" is the transformation that adds a new attribute to an entity in the integrated schema (a similar transformation is called "add" by Motro and Buneman [1981]). For example, the attribute Category for the class Student in the integrated schema may be used to distinguish among Graduate Students (the students represented in View 1) and Undergraduate Students (the students represented in View 2).

Note that the variety of interschema properties is strongly related to the repertory of schema constructs at the disposal of the data model. Among the semantic models, Wiederhold and ElMasri [1979] provide the richest set of interschema properties in the form of various subsets among different schema constructs. For every meaningful pair of constructs in their model, they show an exhaustive list of cases and show how to integrate each by adding interschema properties. Among the relational model based approaches, the richest set of interschema properties—inclusion, exclusion, and union functional dependencies—are provided by Casanova and Vidal [1983] and more recently in the extension of this methodology by Biskup and Convent [1986].

2.8.2 Minimality

In most of the methodologies, the objective of minimality is to discover and eliminate redundancies. A different approach is followed by Batini and Lenzerini [1984],

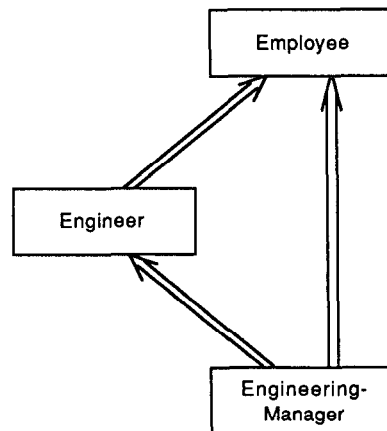
Table 8. Interschema Properties

Reference	Interschema properties
Al-Fedaghi and Scheuermann [1981]	—
Batini and Lenzerini [1984]	Subsetting Generalization Relationship
Casanova and Vidal [1983]	Inclusion dependencies Exclusion dependencies Union functional dependencies
Dayal and Hwang [1984]	Subsetting Subfunction
ElMasri et al. [1987]	Assertions related to extensions Clustering of attributes into classes
Kahn [1979]	—
Motro and Buneman [1981]	Subsetting
Mannino and Effelsberg [1984a]	Generalization Overlap and nonoverlap Attribute scope and meaning
Navathe and Gadgil [1982]	Categorization Subsetting Partitioning
Teorey and Fry [1982]	Generalization Aggregation
Yao et al. [1982]	—
Wiederhold and ElMasri [1979]	Subsetting

where it is stated that discovering the redundancies is a task of conceptual design, whereas their elimination has to be performed during the implementation design phase.

We motivate the minimality notion in Figure 13. There, three subset relationships are present, indicated by double-lined arrows; each arrow points from a subentity to a superentity.

The subset relationship between Engineering-manager and Employee is redundant since it can be derived from the other two. Keeping a minimal number of concepts in the global schema implies dropping the redundant relationship from it. Other typical situations sought are cycles of relationships, derived attributes [Batini and Lenzerini 1984; ElMasri et al. 1987; Navathe and Gadgil 1982], and composition of functions [Yao et al. 1982]. In the relational-model-based approaches, redundancies are related to derived dependencies of various types [Al-Fedaghi and Scheuer-

**Figure 13.** A schema with redundancy.

mann 1981; Casanova and Vidal 1983]. For these approaches, minimality is the driving force behind integration.

As seen from Table 7, most of the schema transformations during restructuring are geared for a removal of redundancy.

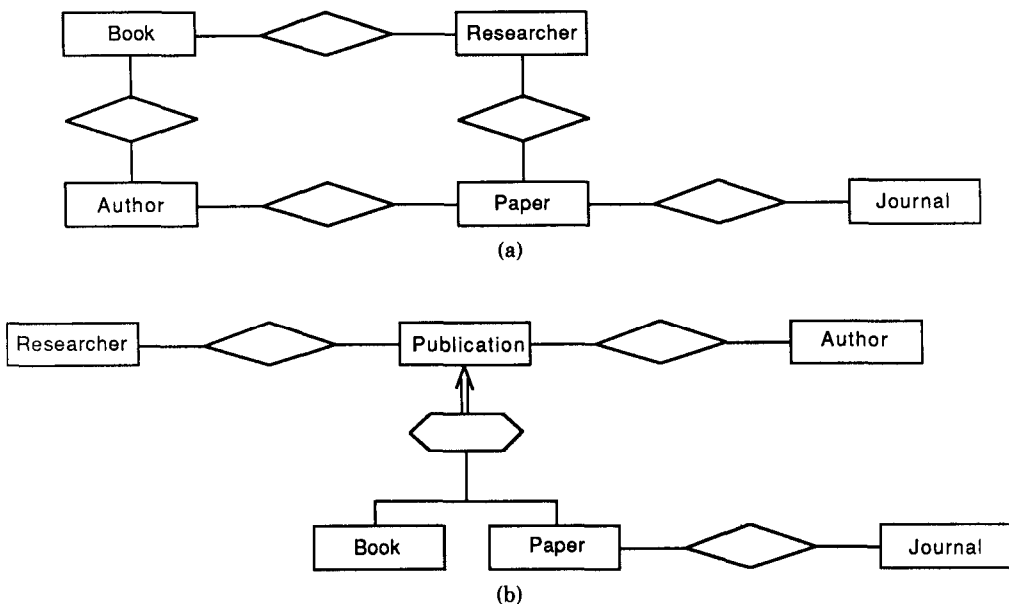


Figure 14. Improving understandability. (a) Schema A. (b) Schema B.

2.8.3 Understandability

Attention to the issue of understandability is diffused in all methodologies. The problem is addressed explicitly by Batini and Lenzerini [1984]. We reproduce an example in Figure 14, where they argue on qualitative terms that, while the two schemas are equivalent, Schema B is more understandable than Schema A. Schema B was obtained from Schema A by adding a generalization hierarchy relating Publication to Book and Paper. In general, for improving understandability, additional schema transformations are needed.

At present, to our knowledge, no quantitative and objective measures of conceptual understandability exist that can be applied here. If a graphical representation of the conceptual model is supported, the shape of the diagram, the total length of connections, the number of crossings and bends, and so forth may be used as parameters to define graphic understandability [Batini et al. 1986].

A specific activity performed during the restructuring step by database integration methodologies is query modification. We already have indicated in Figure 2 that the

mapping of queries is an output of the database integration process. Dayal and Hwang [1984] develop query modification algorithms for modifying global queries into essential local subqueries with duplicate elimination.

3. CONCLUSIONS AND FUTURE WORK

3.1 General Remarks

A few general remarks about the methodologies are in order. The methodologies surveyed can be reviewed on the basis of some general criteria as follows.

3.1.1 Use

Most methodologies were developed as parts of research projects with low emphasis on developing full-scale automated systems. It is obvious that design tools can be built using the concepts from individual methodologies. If the size of the problem can be contained within manual means, however, methodologies also can be used manually.

Partial implementation of some of the methodologies (e.g., Batini and Lenzerini

[1984], Teorey and Fry [1982], and Yao et al. [1982]) have been reported. Nothing has been reported, however, on the actual use of these methodologies to perform view integration.

The entity-relationship model, which provides a basis for the [Batini and Lenzerini 1984; ElMasri et al. 1987; Kahn 1979] methodologies, was reported to be the most widely used model in practice. Chilson and Kudlac [1983] report that the Navathe and Schkolnick model [Navathe and Schkolnick 1978], used in the Navathe and Gadgil [1982] methodology, was also known to the users surveyed.

Out of the methodologies for database integration, that of Dayal and Hwang [1984] has been used with modifications within the framework of the Multibase project at Computer Corporation of America. The Multibase system [Landers and Rosenberg 1982] has been designed and implemented to allow users access to heterogeneous databases in a single system. Several researchers [Hubbard 1980; Chiang et al. 1983; Data Designer 1981; Ferrara 1985] describe tools that allow an integration capability to a limited extent.

3.1.2 Completeness and Detailed Specification

Most of the surveyed methodologies do not provide an algorithmic specification of the integration activities, and they rarely show whether the set of conflicts or the set of transformations considered is complete in some sense. What they provide are general guidelines and concepts on different steps. Methodologies that address well-defined problems of logical design based on purely mechanized procedures such as Al-Fedaghi and Scheuerman [1981], Casanova and Vidal [1983], and Yao et al. [1982] are able to construct precise algorithms. But by their very nature, they cover more of the implementation design compared to conceptual design (according to Table 2).

A side effect of the above problem is that there is no easy way to guarantee convergence in these methodologies, especially for those involving looping structures (see Table 5). The termination of the loops is essentially left to the designer's discretion.

3.2 Missing Aspects

Several aspects are currently missing in methodologies for view integration.

(a) *Processing Specifications in View Integration.* This is the specification of the queries and transactions on component schemas. An initial position reported on view integration in the database design literature [Yao et al. 1982] was that a view integration methodology should have two goals with respect to processing specification:

(1) *Feasibility.* The integrated schema supports any processes on the component schema.

(2) *Performance.* The integrated schema is "optimal" with respect to a given set of component schema queries and transactions. Specifications of queries and transactions are not explicitly used in any methodologies except that of Yao et al. [1982], where an "optimal" structure is selected on the basis of a given set of transactions. We believe that performance analysis based on processing specifications is not meaningful at the conceptual design level since no reasonable performance predictions can be made. Such performance analysis is meaningful only when logical and physical schemas are fully defined in a DBMS. On the other hand, we stress that the real performance measures of conceptual schemas are the goals that we stated in Section 1.3, namely, completeness and correctness, minimality, and understandability.

(b) *Behavioral Specification.* This is the specification of the dynamic properties of objects in the schema (e.g., the value of the salary of an employee can never decrease).

None of the methodologies surveyed model behavioral properties fully. The models adopted by ElMasri et al. [1987] and Navathe and Gadgil [1982] allow them to formulate limited types of behavioral properties in the form of insertion/deletion constraints.

(c) *Schema Mappings*. To support the local views (i.e., external schemas according to the ANSI/SPARC [Klug and Tsichritzis 1977] terminology) of the users of component schemas on the basis of the integrated schema is a problem that is well addressed by the database integration methodologies [Dayal and Hwang 1984; Motro and Buneman 1981]. However, it is only hinted at by ElMasri et al. [1987], Mannino and Effelsberg [1984a], and Navathe and Gadgil [1982] in the form of recognizing “mapping rules” as an output of integration. Only Wiederhold and ElMasri [1979] have given a complete set of rules to support component schemas. Actually, various levels of mappings need to be addressed in going from (or building) external schemas of the integrated schema to (from) one or more external schema(s) of the component schemas.

3.3 Future Research Directions

From Sections 3.1 and 3.2, it is obvious that more work is required on incorporating processing specifications, behavior modeling, and schema mapping in the schema integration methodologies. More research is also required to settle open issues such as the choice of data models and levels of integrity constraint specification. Along with these, the following directions for future research are important.

3.3.1 Extension of Integration Methodologies

View integration methodologies need to be extended to be used in distributed database design. This would imply enriching the inputs by adding more information on the distribution preference of users as well as distributed processing requirements. The principle behind the integration process would remain practically unaltered, but a new set of problems would have to be considered in terms of materializing the so-called local conceptual schemas.

Another possible extension could be to address the design of databases with special properties, such as scientific and statistical databases and databases for computer-

aided design (CAD). In the first case, the integration methodology has to deal with data at different levels of summarization. This leads to a greater complexity of the semantic analysis, an accompanying increase in conflicts, and a corresponding increase in the complexity of conflict-resolution strategies. In the case of CAD databases, problems arise as a result of multiple representations of the same data, as in very large scale integration design, top-down organization of design data, and the far-reaching update propagation.

The statistical and CAD databases are often subjected to database integration for allowing sharing of information. New methodologies of database integration for such cases need to be designed; the existing works seem limited in this area.

3.3.2 Knowledge Base Integration

Integration of knowledge bases has received attention in the literature only recently [Eick and Lockemann 1985]. Knowledge bases treat classes and their instances together: This implies that data and metadata coexist in the representation. Moreover, they provide richer linguistic mechanisms: Knowledge is often expressed in the form of logical assertions, productions, and inference rules. Rule integration is a problem in itself. These considerations bring a new set of issues that are not covered presently in the surveyed methodologies.

3.3.3 Process Integration

This refers to the activity of integrating and transforming a set of processes applicable to component schemas into a set of processes applicable to the integrated schema. It seems that many notions used for data schema integration can be transferred to process integration: For example, the goals (Section 1.3) are equally applicable; so are the concepts of equivalence, semantic analysis, conflict detection and resolution, and transformations. Tucher et al. [1985] consider database design to be an integration of process modules. Some preliminary work is under way on the related problem of program transformation [Demo 1983; Demo and Kundu 1985].

3.3.4 Expert Systems for Schema Integration

As pointed out in the Introduction and Section 1, schema integration is a difficult and complex task. An expert system approach to database design in general and to schema integration in particular on the basis of the rules and heuristics of design is worth investigating. Projects have already been under way in this area (e.g., see Bouzeghoub et al. [1986] and Shin and Irani [1985]). Model dependent rules should be used in the comparison and conforming activities with the goal of improving the equivalence and/or compatibility of component schemas. Alternative schema transformations can be suggested or evaluated by the system when a conflict must be solved. Selection among alternative schemas for integration can be guided by system-designer interaction.

APPENDIX 1. A SUMMARY DESCRIPTION OF METHODOLOGIES

In the following, the methodologies surveyed in this paper are briefly described. The same categories of description are used for each methodology. These descriptions should only be treated as a quick reference guide and not as a substitute for the original descriptions of the methodologies. They are included here to highlight the fact that, although the general intent of all methodologies is very similar, the actual mechanics vary greatly. The terminology of the authors is used without modification. Words in parentheses refer to equivalent terms used in this paper.

Of the above methodologies surveyed, those of Dayal and Hwang [1984], Mannino and Effelsberg [1984a], and Motro and Buneman [1981] apply to database integration; the method of [ElMasri et al. [1987] is used for database integration as well as view integration, whereas the remaining methodologies apply to view integration only.

Al-Fedaghi and Scheuermann [1981]

Type: View integration methodology.

Model: Relational model.

Input: n external views (component schemas), given in terms of relations and functional dependencies.

Output: n external schemas, one conceptual schema (integrated schema), a mapping mechanism between external schemas and conceptual schema.

Processing specifications considered: No.

Integration strategy:

- (1) Find sets of functional dependencies common to some set of external views.
- (2) Eliminate in previous sets (local) redundant dependencies.
- (3) Remove redundant dependencies due to transitivity in the global set of dependencies, thus producing a nonredundant cover of the conceptual schema.
- (4) Identify dependencies that were eliminated in previous steps, but must now be readded to external views in order to minimize their effect on the mapping process; construct external views.

Special features:

- (1) The main goal of the methodology is to obtain mappings that
 - (a) preserve compatibility between relations and dependencies in external schemas and in the integrated schema;
 - (b) reduce interferences between insert/delete operations in different external schemas.
- (2) The methodology assures that all relations are projections of a universal relation.

Batini and Lenzerini [1984]

Type: View integration methodology.

Model: Entity-relationship model (see Appendix 2).

Input:

user schemata (component schemas), enterprise schema, weights for schemata.

Output: Global schema (integrated schema).

Processing specifications considered: No.

Integration strategy:

Integration strategy:

- (1) Choose the enterprise schema as the base schema.
- (2) While new schemas are to be integrated, do
 - (2.1) Choose a new schema.
 - (2.2) Find conflicts between the two schemas.
 - (2.3) Amend the two schemas in order to conform them.
 - (2.4) Merge the schemas.
 - (2.5) Analyze the draft integrated schema in order to discover redundancies and simplify the representation.

- (1) Combine user views, merging relation schemas of the two different views and defining new inclusion, exclusion, and union functional dependencies.
- (2) Optimize the temporary conceptual schema, trying to minimize redundancy and the size of the schema.

Special features:

Special features:

- (1) Several indications are suggested to guide the designer in the investigation of conflicts (e.g., type inconsistencies, concept likeness/unlikeness).
- (2) For every indication, several scenarios are proposed (i.e., typical solutions of the conflict).
- (3) Several types of equivalence transformations are supplied to confirm the representation of concepts.
- (4) A specific activity is suggested to improve understandability of the global schema.

- (1) The relational model is enriched with interrelational dependencies useful for expressing how data in distinct views are interrelated.
- (2) It is assumed that a preliminary integration process has been carried out to detect which structures of different views represent the same information and interrelational dependencies.
- (3) The optimization procedure (Step 2) is shown to be correct for a special class of schemas, called restricted schemas; a restricted schema is essentially a representation of a collection of entities-relationships, identified by their keys.

Related references: Batini and Lenzerini [1983] and Batini et al. [1983].

Dayal and Hwang [1984]

Casanova and Vidal [1983]

Type: View integration methodology.

Type: Database integration methodology.

Model: Relational model. Besides functional dependencies, other types of dependencies are considered: inclusion, exclusion, and union functional dependencies.

Model: Functional model. The model uses two constructs: entities and functions (i.e., properties of entities, relationships among entities). Functions may be single valued or multivalued. Entities may be user defined (e.g., Person) or else constants (e.g., Boolean). A generalization abstraction is provided among entities and functions.

Input: Two user views (component schema).

Input:

local schemas of existing databases, queries.

Output: Conceptual schema (integrated schema).

Output:

global interface to databases, modified queries.

Processing specifications considered: No.

Processing specifications considered: Queries.

Integration strategy:

- (1) Solve conflicts among concepts in local schemas (naming, scale, structural, abstraction conflicts).
- (2) Solve conflicts among data in existing databases (inconsistencies in identifiers, different degree of obsolescence, different degree of reliability).
- (3) Modify queries and make them consistent with the global schema.

Special features:

- (1) Generalization abstraction is uniformly used as a means to combine entities and resolve different types of conflicts.
- (2) A detailed algorithm is given for query modification and is formally proved correct and nonredundant by Hwang [1982].

ElMasri et al. [1987]

Type: Both view integration in logical database design and database integration.

Model: Entity-Category-Relationship (ECR) model [ElMasri et al. 1985], which recognizes, besides entities and relationships, the concept of categories. Categories are used for two purposes: to show a generalization of a superentity into subentities and to simply allow for the definition of a subset of an entity based on some predicate.

Input:

n schemas, which represent either user views or existing databases represented in the ECR model;
attribute equivalence assertions;
object class extension assertions.

Output:

integrated schema,
mappings between integrated and conceptual schemas.

Processing specifications considered: Not to determine the result of integration. However, the problem of dealing with queries on the integrated schema is addressed in terms of mappings.

Integration strategy:

- (1) Transform existing schemas into ECR if needed.
- (2) Preintegration, which consists of an interleaved application of schema analysis and modification with assertion specification.
- (3) Integration of object classes.
- (4) Integration of relationship classes.
- (5) Generation of mappings.

The above procedure is followed as an *n*-ary integration process.

Special features:

- (1) A very detailed treatment of attribute and object extension assertions via consistency checking and verification of algorithms is included.
- (2) The methodology uses the notion of extension of attribute types and object classes as a basis for comparison.
- (3) The methodology applies equally to view integration and database integration.

Related references: ElMasri and Navathe [1984], Larson et al. [1986], Navathe et al. [1984], Navathe et al. [1986], Weeldreyer [1986].

Kahn [1979]

Type: View integration methodology.

Model: Entity-relationship model (see Appendix 2).

Input: Local information structures (component schemas).

Output: Global information structure (integrated schema).

Processing specifications considered: No.

Integration strategy:

- (1) (Entity step) Aggregate entities.
 - (1.1) Standardize names.
 - (1.2) Aggregate entities to form a non-redundant collection.
 - (1.3) Check entities against processing requirements.
 - (1.4) Eliminate nonessential attributes.
 - (1.5) Simplify the representation.

- (2) (Relationship step) Aggregate relationships.
 - (2.1) Standardize names.
 - (2.2) Analyze consistency of relationship cardinalities versus entity cardinalities.
 - (2.3) Aggregate relationships.
 - (2.4) Determine conditional and existence-dependent relationship.
 - (2.5) Eliminate all redundant relationships.

Special features:

- (1) A rich set of heuristics is suggested to guide the designer in discovering conflicts.
- (2) Several types of qualities are defined for the integrated schema, and strategies are suggested to achieve these.

Mannino and Effelsberg [1984a]

Type: Database integration.

Model: Generalized entity manipulator.

Input: Local schemas in a common data model, interschema assertions about entity types and attributes.

Output: Global view objects, global view mapping, integration schema objects.

Processing specifications considered: No.

Integration strategy:

- (1) Transform each local schema into an easy-to-integrate form.
- (2) Match the entity types and attributes of the local schemas.
- (3) Define assertions about the entity types that can be generalized and then define assertions about equivalent attributes.
- (4) Merge pairs of "generalizable" entity families as indicated by the assertions and designer preferences.
- (5) Define global attribute formats and conversion rules for the global entity types.

Special features:

- (1) The merging step uses entity families (collection of entity types related by generalization) rather than simple entity types.

- (2) Companion global view definition language that uses the same set of integration operators as the methodology.
- (3) Semantic equivalence and range of meaning of individual attributes, groups of attributes, and functions of attributes can be defined in attribute assertions.

Steps 2, 3, and 4 may be performed in sequence or iteratively with backtracking.

Related references: Mannino and Effelsberg [1984b], Mannino and Karle [1986], and Mannino et al. [1986].

Motro and Buneman [1981]

Type: Database integration methodology.

Model: Functional model. Constructs of the model are classes of objects, which may be related by two types of functions—att, by which one class becomes an attribute of another class, and gen, which establishes a generalization relationship.

Input:

Two logical (component) schemas with the corresponding databases, queries.

Output:

superview (global schema), modified queries.

Processing specifications considered: Queries.

Integration strategy:

- (1) Merge the two (independent) logical schemas by combining initially primitive classes.
- (2) While new restructurings can be applied to the temporary integrated schema, do
 - (2.1) Choose a restructuring primitive and apply to the integrated schema.

Special features:

- (1) The main feature of the methodology is to provide a large and powerful set of restructuring primitives while no



heuristics are given to discipline their use.

Related references: Motro [1981].

Navathe and Gadgil [1982]

Type: View integration methodology.

Model: Navathe-Schkolnick model. The main construct of the model is the object (type), representing either an entity or an association, which can be recursively defined in terms of entities or associations. Other concepts are connectors, which model insertion/deletion properties of associations and subsets between objects. Associations are divided into three types: subsetting, partitioning, and categorizing associations.

Input:

enterprise view,
local views (component schemas),
integration strategy,
interview and intraview assertions,
processing requirements.

Output:

global view,
mapping rules,
unresolved conflicts,
modified assertions.

Integration strategy:

- (1) Divide views into classes of equivalent views, identical views, single views.
- (2) Integrate classes checking for conflicts (among names, keys, etc.) and solving them on the basis of assertions and order of preference.
- (3) While new view assertion operations are applicable, do
 - (3.1) Perform new integrations between intermediate and semi-integrated views in a way similar to Step 2.
- (4) Generate mapping rules determining how each of the component views can be obtained from the integrated view.

Special features:

- (1) Equivalence and containment relations among information contents of user schemas are assumed as input to the design.
- (2) A taxonomy is given for types of comparisons among objects, conflicts, and view integration operations.
- (3) Conflicts are generally resolved by adopting the most restrictive specification.
- (4) Attention is given to the problem of automating the view integration process, distinguishing activities that can be solved automatically and activities that ask for interaction with the designer/user.

Teorey and Fry [1982]

Type: View integration methodology.

Model: Semantic hierarchical model. Constructs are classes of objects, aggregation abstractions among objects by which an object is seen as an abstraction of the set of its properties, and generalization abstractions.

Input:

information perspective (component schemas),
application perspective,
event perspective,
corporate perspective,
policy guidance and rules.

Output:

global information structure (integrated schema),
conflicts.

Processing specifications considered: No.

Integration strategy:

- (1) Order local views as to importance with respect to the specific design objectives.
- (2) Consolidate local views, two at a time (the order is determined by Step 1).
- (3) Solve conflicts that have arisen in Step 2.

Special features:

- (1) Attention is given to the problem of integration of processing specifications, but no specific strategies and methods are proposed.
- (2) Different types of integration processing strategies (see Section 2.5) are compared. The binary balanced strategy is claimed to be the most effective.

Wiederhold and ElMasri [1979]

Type: View integration methodology.

Model: Structural model. Such a model is constructed from relations that are used to represent entity classes and several types of relationships among entity classes. Other types of relationships are represented by connections between relations.

Input: Two data models (component schemas).

Output:
Integrated database model (integrated schema),
database submodels.

Processing specifications considered: Only primitive operations on concepts (insertion, deletion).

Integration strategy:

- (1) Find all compatible pairs of relations.
- (2) For each pair of relations, integrate the connection between them.
- (3) Integrate compatible relations.

Special features:

- (1) Owing to the rich variety of modeling constructs of the structural model, an extensive set of conflicts is presented and analyzed, and solutions are provided.
- (2) Mapping rules are derived from the integration process to express data models consistently with the integrated database model.

Related references: ElMasri [1980] and ElMasri and Wiederhold [1979].

Yao et al. [1982]

Type: View integration methodology.

Model: Functional model. Constructs of the model are nodes, classified into simple nodes representing atomic data elements and tuple nodes, representing nonfunctional (i.e., many-to-many) relationships among nodes, and functions among nodes.

Input: Two schemas.

Output: The integrated schema.

Processing specifications considered: Yes, in language TASL.

Integration strategy:

- (1) Merge nodes with same values.
- (2) Merge nodes that are subsets of other nodes.
- (3) Remove redundant functions and modify corresponding transaction specifications.

Special features:

- (1) The main aspect dealt with in the methodology is to determine and remove redundancy.
- (2) Paths to be removed are found by using processing specification information.
- (3) A transaction specification language (TASL) accompanies the methodology.

APPENDIX 2. THE ENTITY-RELATIONSHIP MODEL

The original model, known as the entity-relationship Model (E-R), was proposed by Chen [1976]. Further extensions of the model appear in Dos Santos et al. [1980] and Scheuermann et al. [1980]. The following concepts are defined in the model.

An *entity* is a class of objects of the real world having similar characteristics and

properties. A *relationship* is a class of elementary facts or associations among entities. An attribute is an elementary property either of an entity or a relationship. An entity E_1 is a *subset* of an entity E_2 if every object belonging to E_1 also belongs to E_2 . An entity E is a *generalization* of entities E_2, E_2, \dots, E_n if

- (1) every E_i is a subset of E , and
- (2) every object belonging to E belongs exactly to one of the E_i 's.

A diagrammatic representation is widely used with the E-R model. In Table 9 we show the correspondence between the concepts of the model and the diagrammatic symbols.

An example of a schema appears in Figure 15, which describes information of interest in a data processing department of a company.

The information is about employees (which includes programmers, managers, and senior programmers), projects, and languages. The entities and their corresponding attributes are as follows:

- Employee: Employee#, Last_name, Age
- Project: Project#, Name
- Language: Name, Version
- Manager: Budget
- Programmer: none
- Senior_programmer: Years_of_experience

The relationships among the above entities are:


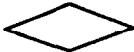


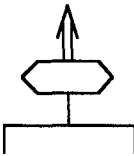
“Works_on,” connecting Employee and Projects.

“Uses,” connecting Programmer, Project and Language.

The Works-on relationship has an attribute %_of_time.

Senior-Programmer is a subset of Programmer, and Employee is a generalization of Programmer and Manager. The resulting hierarchy among Employee as a generic entity and Programmer and Manager as its specialized entities is denoted by the name Rank. Note that, by virtue of the generalization hierarchies, Manager and Program-

Table 9. Concepts of the Entity-Relationship Model and Corresponding Symbols

CONCEPT	SYMBOL
Entity	
Relationship	
Attribute	
Subset	
Generalization	

mer inherit all properties (attributes and relationship types) of Employee, which include attributes of Employee and relationship “Works_on.” Owing to the subset relationship, Senior_programmer inherits all the properties of Programmer, which include relationship “Uses” and all attributes from Employee.

Various types of constraints have been specified to go with the E-R model. Here we only refer to the cardinality constraints. The cardinality constraint restricts the number of relationships in which a specific entity can participate. In the example, the cardinality constraint governing the “Works On” relationship is many to many ($m:n$); that is, an employee may Work on many projects, and a project may have many employees who Work on it. Common cardinality constraints are: one to one (1:1), one to many (1:n), many to one ($n:1$), and many to many ($m:n$).

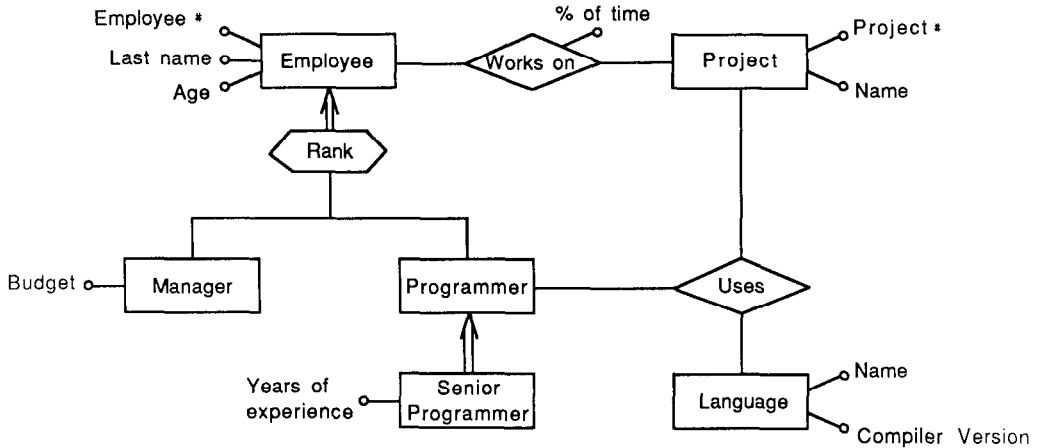


Figure 15. An entity-relationship schema.

ACKNOWLEDGMENTS

Navathe's research was partly supported by National Science Foundation grant No. INT-8400216. Batini's and Lenzerini's research was partly supported by Progetto Finalizzato Informatica and Progetto Finalizzato Trasporti, CNR, Italy.

We gratefully acknowledge the patience and cooperation of Sharon Grant and Claudio Dollari in preparing this manuscript. The comments of anonymous referees were very helpful in revising an earlier draft of the paper.

REFERENCES

A. Complete Methodologies for Schema Integration

AL-FEDAGHI, S., AND SCHEUERMANN, P. 1981. Mapping considerations in the design of schemas for the relational model. *IEEE Trans. Softw. Eng. SE-7*, 1 (Jan.).

BATINI, C., AND LENZERINI, M. 1984. A methodology for data schema integration in the entity relationship model. *IEEE Trans. Softw. Eng. SE-10*, 6 (Nov.), 650-663.

CASANOVA, M., AND VIDAL, M. 1983. Towards a sound view integration methodology. In *Proceedings of the 2nd ACM SIGACT/SIGMOD Conference on Principles of Database Systems* (Atlanta, Ga., Mar. 21-23). ACM, New York, pp. 36-47.

DAYAL, U., AND HWANG, H. 1984. View definition and generalization for database integration in multibase: A system for heterogeneous distributed databases. *IEEE Trans. Softw. Eng. SE-10*, 6 (Nov.), 628-644.

ELMASRI, R., LARSON, J., AND NAVATHE, S. B. 1987. Integration algorithms for federated databases and logical database design. Tech.

Rep., Honeywell Corporate Research Center (submitted for publication).

KAHN, B. 1979. A structured logical data base design methodology. Ph.D. dissertation, Computer Science Dept., Univ. of Michigan, Ann Arbor, Mich.

MANNINO, M. V., AND EFFELBERG, W. 1984a. A methodology for global schema design, Computer and Information Sciences Dept., Univ. of Florida, Tech. Rep. No. TR-84-1, Sept.

MOTRO, A., AND BUNEMAN, P. 1981. Constructing superviews. In *Proceedings of the International Conference on Management of Data* (Ann Arbor, Mich., Apr. 29-May 1). ACM, New York.

NAVATHE, S. B., AND GADGIL, S. G. 1982. A methodology for view integration in logical data base design. In *Proceedings of the 8th International Conference on Very Large Data Bases* (Mexico City). VLDB Endowment, Saratoga, Calif.

TEOREY, T., AND FRY, J. 1982. *Design of Database Structures*. Prentice-Hall, Englewood Cliffs, N.J.

WIEDERHOLD, G., AND ELMASRI, R. 1979. A structural model for database systems. Rep. STAN-CS-79-722, Computer Science Dept., Stanford Univ., Stanford, Calif.

YAO, S. B., WADDLE, V., AND HOUSEL, B. 1982. View modeling and integration using the functional data model. *IEEE Trans. Softw. Eng. SE-8*, 6, 544-553.

B. Related Work

ALBANO, A., CARDELLI, L., AND ORSINI, R. 1985. Galileo: A strongly typed, interactive conceptual language. *ACM Trans. Database Syst. 10*, 2 (June), 230-260.

ATZENI, P., AUSIELLO, G., BATINI, C., AND MOSCARINI, M. 1982. Inclusion and equivalence between relational database schemata. *Theor. Comput. Sci. 19*, 267-285.

- BATINI, C., AND LENZERINI, M. 1983. A conceptual foundation to view integration. In *Proceedings of the IFIP TC.2 Working Conference on System Description Methodologies* (Kecskmet, Hungary). Elsevier, Amsterdam, pp. 109-139.
- BATINI, C., LENZERINI, M., AND MOSCARINI, M. 1983. Views integration. In *Methodology and Tools for Data Base Design*, S. Ceri, Ed. North-Holland, Amsterdam.
- BATINI, C., DEMO, B., AND DI LEVA, A. 1984. A methodology for conceptual design of office data bases. *Inf. Syst.* 9, 3, 251-263.
- BATINI, C., NARDELLI, E., AND TAMASSIA, R. 1986. A layout algorithm for data flow diagrams. *IEEE Trans. Softw. Eng. SE-12*, 4 (Apr.), 538-546.
- BEERI, C., BERNSTEIN, P., AND GOODMAN, N. 1978. A sophisticate's introduction to database normalization theory. In *Proceedings of the 4th International Conference on Very Large Data Bases* (West Berlin, Sept. 13-15). IEEE, New York.
- BERNSTEIN, P. A. 1976. Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst.* 1, 4 (Dec.), 277-298.
- BILLER, H. 1979. On the equivalence of data base schemas: A semantic approach to data translation. *Inf. Syst.* 4, 1, 35-47.
- BILLER, H., AND NEUHOLD, E. J. 1982. Concepts for the conceptual schema. In *Architecture and Models in Data Base Management Systems*, G. M. Nijssen, Ed. North Holland, Amsterdam, pp. 1-30.
- BISKUP, J., AND CONVENT, B. 1986. A formal view integration method. In *Proceedings of the International Conference on the Management of Data* (Washington, D.C., May 28-30). ACM, New York.
- BISKUP, J., DAYAL, U., AND BERNSTEIN, P. A. 1979. Independent database schemas. In *Proceedings of the International Conference on the Management of Data* (Boston, Mass., May 30-June 1). ACM, New York.
- BOUZEGHOUB, M., GARDARIN, G., AND METAIS, E. 1986. Database design tools: An expert systems approach. In *Proceedings of 11th International Conference of Very Large Databases* (Stockholm, Sweden). Morgan Kaufmann, Los Altos, Calif.
- BRODIE, M. L. 1981. On modelling behavioural semantics of data. In *Proceedings of the 7th International Conference on Very Large Data Bases* (Cannes, France, Sept. 9-11). IEEE, New York, pp. 32-41.
- BRODIE, M. L., AND ZILLES, S. N., Eds. 1981. In *Proceedings of the Workshop on Data Abstraction, Databases, and Conceptual Modelling*. SIGPLAN Not. 16, 1 (Jan.).
- CARSWELL, J. L., AND NAVATHE, S. B. 1986. SA-ER: A methodology that links structured analysis and entity relationship modeling for database design. In *Proceedings of the 5th International Conference on the Entity Relationship Approach*, S. Spaccapietra, Ed. (Dijon, France, Nov.), pp. 19-36.
- CERI, S., Ed. 1983. *Methodology and Tools for Database Design*. North-Holland, Amsterdam.
- CERI, S., AND PELAGATTI, G. 1984. *Distributed Databases: Principles and Systems*. McGraw-Hill, New York.
- CERI, S., PELAGATTI, G., AND BRACCHI, G. 1981. A structured methodology for designing static and dynamic aspects of data base applications. *Inf. Syst.* 6, 1, 31-45.
- CHEN, P. P. 1976. The entity-relationship model—Toward a unified view of data. *ACM Trans. Database Syst.* 1, 1 (Mar.), 9-36.
- CHEN, P. P. 1983. English sentence structure and entity-relationship diagrams. *J. Inf. Sci.* 29, 127-150.
- CHIANG, W., BASAR, E., LIEN, C., AND TEICHROEW, D. 1983. Data modeling with PSL/PSA: The view integration system (VIS). ISDOS Rep. No. M0549-0, Ann Arbor, Mich.
- CHILSON, D., AND KUDLAC, C. 1983. Database design: A survey of logical and physical design techniques. *Database* 15, 1 (Fall).
- DATA DESIGNER 1981. Data designer product description. Database Design Inc., Ann Arbor, Mich.
- DEMO, B. 1983. Program analysis for conversion from a navigation to a specification database interface. In *Proceedings of the 9th International Conference on Very Large Data Bases* (Florence, Italy). VLDB Endowment, Saratoga, Calif.
- DEMO, B., AND KUNDU, S. 1985. Modeling the CO-DASYL DML execution context dependency for application program conversion. In *Proceedings of the International Conference on Management of Data* (Austin, Tx., May 28-30). ACM, New York, pp. 354-363.
- DOS SANTOS, C. S., NEUHOLD, E. J., AND FURTADO, A. L. 1980. A data type approach to the entity relationship model. In *Proceedings of the International Conference on the Entity Relationship Approach to System Analysis and Design*, P. Chen, Ed. (Los Angeles, 1979). North-Holland, Amsterdam, pp. 103-120.
- EICK, C. F., AND LOCKEMANN, P. C. 1985. Acquisition of terminological knowledge using database design techniques. In *Proceedings of the International Conference on Management of Data* (Austin, Tx., May 28-30). ACM, New York, pp. 84-94.
- ELMASRI, R. 1980. On the design, use and integration of data models. Ph.D. dissertation, Rep. No. STAN-CS-80-801, Dept. of Computer Science, Stanford Univ., Stanford, Calif.
- ELMASRI, R., AND NAVATHE, S. B. 1984. Object integration in database design. In *Proceedings of the IEEE COMPDEC Conference* (Anaheim, Calif., Apr.). IEEE, New York, pp. 426-433.
- ELMASRI, R., AND WIEDERHOLD, G. 1979. Data model integration using the structural model. In

- Proceedings of the International Conference on Management of Data* (Boston, Mass., May 30–June 1). ACM, New York.
- ELMASRI, R., WEELDRYER, J., AND HEVNER, A. 1985. The category concept: An extension to the entity-relationship model. *Data Knowl. Eng.* 1, 1 (June).
- FERRARA, F. M. 1985. EASY-ER: An integrated system for the design and documentation of data base applications. In *Proceedings of the 4th International Conference on the Entity Relationship Approach* (Chicago, Ill.). IEEE Computer Society, Silver Spring, Md., pp. 104–113.
- HAMMER, M., AND MCLEOD, D. 1981. Database description with SDM: A semantic database model. *ACM Trans. Database Syst.* 6, 3 (Sept.), 351–386.
- HUBBARD, G. 1980. *Computer Assisted Data Base Design*. Van Nostrand-Reinhold, New York.
- HWANG, H. Y. 1982. Database integration and optimization in multidatabase systems. Ph.D. dissertation, Dept. of Computer Science, Univ. of Texas, Austin, Oct.
- KLUG, A., AND TSICHRITZIS, D., Eds. 1977. The ANSI/X3/SPARC Report of the Study Group on Data Base Management Systems. AFIPS Press, Reston, Va.
- LANDERS, T. A., AND ROSENBERG, R. L. 1982. An overview of Multibase. In *Distributed Databases*, H. J. Schneider, Ed. North-Holland, Amsterdam.
- LARSON, J., NAVATHE, S. B., AND ELMASRI, R. 1986. Attribute equivalence and its role in schema integration. Tech. Rep., Honeywell Computer Sciences Center, Golden Valley, Minn.
- LUM, V., GHOSH, S., SCHKOLNICK, M., JEFFERSON, D., SU, S., FRY, J., AND YAO, B. 1979. 1978 New Orleans data base design workshop. In *Proceedings of the 5th International Conference on Very Large Data Bases* (Rio de Janeiro, Oct. 3–5). IEEE, New York, pp. 328–339.
- MAIER, D. 1983. *The Theory of Relational Databases*. Computer Science Press, Potomac, Md.
- MANNINO, M. V., AND EFFELSBERG, W. 1984b. Matching techniques in global schema design. In *Proceedings of the IEEE COMPDEC Conference* (Los Angeles, Calif.). IEEE, New York, pp. 418–425.
- MANNINO, M. V., AND KARLE, C. 1986. An extension of the general entity manipulator language for global view definition. *Data Knowl. Eng.* 2, 1.
- MANNINO, M. V., NAVATHE, S. B., AND EFFELSBERG, W. 1986. Operators and rules for merging generalization hierarchies. Working Paper, Graduate School of Business, Univ. of Texas, Austin, April 1986.
- MCLEOD, D., AND HEIMBIGNER, D. 1980. A federated architecture for data base systems. In *Proceedings of the AFIPS National Computer Conference*, vol. 39. AFIPS Press, Arlington, Va.
- MOTRO, A. 1981. Virtual merging of databases. Ph.D. dissertation, Tech. Rep. #MS-CIS-80-39, Computer Science Dept., Univ. of Pennsylvania, Philadelphia, Pa. 1981.
- MYLOPOULOS, J., BERNSTEIN, P. A., AND WONG, H. K.T. 1980. A language facility for designing database-intensive applications. *ACM Trans. Database Syst.* 5, 2 (June) 185–207.
- NATIONAL BUREAU OF STANDARDS 1982. Data base directions: Information resource management—strategies and tools. Special Publ. 500–92, A. Goldfine, Ed. U.S. Dept. of Commerce, Washington, D.C., Sept. 1982.
- NAVATHE, S.B., AND SCHKOLNICK, M. 1978. View representation in logical database design. In *Proceedings of the International Conference on Management of Data* (Austin, Tex.). ACM, New York, pp. 144–156.
- NAVATHE, S. B., AND KERSCHBERG, L. 1986. Role of data dictionaries in information resource management. *Inf. Manage.* 10, 1.
- NAVATHE, S. B., SASHIDHAR, T., AND ELMASRI, R. 1984. Relationship matching in schema integration. In *Proceedings of the 10th International Conference on Very Large Data Bases* (Singapore). Morgan Kaufmann, Los Altos, Calif.
- NAVATHE, S. B., ELMASRI, R., AND LARSON, J. 1986. Integrating user views in database design. *IEEE Computer* 19, 1 (Jan.), 50–62.
- NG, P., JAJODIA, S., AND SPRINGSTEEL, F. 1983. The problem of equivalence of entity relationship diagrams. *IEEE Trans. Softw. Eng.* SE-9, 5, 617–630.
- OLLE, T. W., SOL, H. G., AND VERRIJN-STUART, A. A., Eds. 1982. Information systems design methodologies: A comparative review. In *Proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies* (Noordwijkerhout, The Netherlands). North-Holland, Amsterdam.
- RISSANEN, J. 1977. Independent components of relations. *ACM Trans. Database Syst.* 2, 4 (Dec.), 317–325.
- ROLLAND, C., AND RICHARDS, C. 1982. Transaction modeling. In *Proceedings of the International Conference on Management of Data* (Orlando, Fla., June 2–4). ACM, New York, pp. 265–275.
- SAKAI, H. 1981. A method for defining information structures and transactions in conceptual schema design. In *Proceedings of the 7th International Conference on Very Large Data Bases* (Cannes, France, Sept. 9–11). IEEE, New York, pp. 225–234.
- SCHUEERMANN, P., SCHIFFNER, G., AND WEBER, H. 1980. Abstraction capabilities and invariant properties modeling within the entity relationship approach. In *Proceedings of the International Conference on Entity Relationship Approach to System Analysis and Design*, P. Chen, Ed. (Los Angeles, 1979). North-Holland, Amsterdam.
- SHIN, D. G., AND IRANI, K. B. 1985. Knowledge-based distributed database system design. In *Proceedings of the International Conference on*

- Management of Data* (Austin, Tex., May 28–30). ACM, New York, pp. 95–105.
- SHIPMAN, D. W. 1980. The functional data model and data language DAPLEX. *ACM Trans. Database Syst.* 6, 1 (Mar.), 140–173.
- SMITH, J. M., AND SMITH, D. C. 1977. Database abstraction: Aggregation and generalization. *ACM Trans. Database Syst.* 2, 2 (June), 105–133.
- TUCHERMAN, L., FURTADO, A. L., AND CASANOVA, M. A. 1985. A tool for modular database design. In *Proceedings of the 11th International Conference on Very Large Data Bases* (Stockholm, Sweden). Morgan Kaufmann, Los Altos, Calif.
- ULLMAN, J. D. 1982. *Principles of Database Systems*, 2nd ed. Computer Science Press, Potomac, Md.
- WEELDREYER, J. A. 1986. Structural aspects of the entity–category–relationship model of data, Tech. Rep. HR-80-251, Honeywell Computer Sciences Center, Golden Valley, Minn.

Received March 1985; final revision accepted December 1986.