Improving Integration Quality for Heterogeneous Data Sources

Inaugural–Dissertation

zur Erlangung des Doktorgrades der
Mathematisch–Naturwissenschaftlichen Fakultät
der Heinrich–Heine–Universität Düsseldorf

vorgelegt von

Evgeniya Altareva
aus Sankt Petersburg, Russland

Düsseldorf
2004

Gedruckt mit der Genehmigung der
Mathematisch–Naturwissenschaftlichen Fakultät
der Heinrich–Heine–Universität Düsseldorf

Referent: Prof. Dr. Stefan Conrad

Korreferent: Prof. Dr. Arndt von Haeseler

Tag der mündlichen Prüfung: 24.01.2005

Acknowledgement

I would like to express my thanks to all the people who supported me in the development of this thesis.

First and foremost, my sincere acknowledgements go to Prof. Dr. Stefan Conrad, my supervisor and the first referee of the thesis. The atmosphere prevailing at his chair, the numerous discussions, that we had during the work on the thesis, his insight, his vision certainly played one of the major roles in making this work accomplished. Also, I would like to thank Prof. Dr. Arndt von Haeseler for his interest in my work and willingness to be a second referee.

Complementary part of the DIAsDEM project is carried out at the Otto–von–Guericke–University of Magdeburg. In this regard I would like to thank the second project leader, Prof. Dr. Myra Spiliopoulou for her enthusiastic support and cooperative work.

My work would have never been so interesting and the atmosphere would have never been so informal without collaboration with my colleagues at the database group. I would like to extend my complements to Cristian Pérez de Laborda, Christopher Popfinger, Johanna Vompras, Mireille Samia, Marga Potthoff and Guido Königstein.

A special word of thanks goes to my former colleagues at the chair of Prof. Dr. Hans–Peter Kriegel at the Ludwig–Maximilians–University of Munich, where I have started to work on my thesis under supervision of Prof. Dr. Stefan Conrad.

Evgeniya Altareva
Düsseldorf, October 2004

Abstract

This work considers a problem of integrating heterogeneous semi–structured data sources with the purpose of estimating integration quality (IQ). During the integration of such data sources the IQ estimation plays an important role, because correspondences and dependencies within and across the sources are not completely known, the schema or semantics might be missing, which leads to results with unpredictable trustworthiness. Therefore, we consider existing methods of analysis of such data sources and investigate a possible scenario of the integration process. We analyze a problem of uncertainty in the integration process. For that we introduce examples demonstrating present inability of accounting for the combined uncertainties affecting integration quality. We introduce a classification of the types of uncertainties. In order to account for the uncertainties we suggest using the statistical method Latent Class Analysis (LCA), related to the Latent Variable Models. This method allows to analyze the influence of the latent factors on the set of data. As related to the task of integration, by a latent factor we understand belonging of an object to a real–world class and in its turn the role of LCA is to interpret correlation of discovering identical objects from different data sources as a display of that universal factor. We build a statistical model of the integration task, i.e., draw correspondences between the terms of statistics and the terms of integration. Presence of at least three data sources is necessary for making use of LCA, at that, when integrating two sources, an integrated database itself can represent a lacking third source. The result of the analysis is the probability value of the real–world class membership for a considered group of objects. Derived by LCA real–world class membership value includes influence of all types of uncertainties and reflects IQ. By applying LCA to each triplet of the corresponding classes at the lowest schema level and obtaining real–world class membership, we can calculate the support of the real–world class for any level of the database, including database itself, as a weighted average of the real–world class membership for all classes at the lowest level. The proposed approach does not solve common problems of integration of the heterogeneous data sources, but rather can be used for evaluating and improving IQ. Capability to evaluate the IQ gives an important tool to the users concerned with the data's trustworthiness. It helps them to answer the

question of whether or not and to what extent they can trust the data and the database queries. In case of unacceptable IQ, by tuning integration parameters, for example, changing integration strategy, appropriate IQ can potentially be achieved.

Contents

1 Introduction

This chapter shortly introduces the context of this thesis, which contributes to the field of database integration, especially to the task of evaluating and improving integration quality. In Section 1.1 we briefly describe the problems that occur during the integration of heterogeneous data sources, giving motivation for our work. Section 1.2 provides an outline of this thesis.

1.1 Motivation

The problem of integration of heterogeneous semi–structured data sources is a subject that attracts various groups of researchers ([BCV99, BM99, CFOA02, IJG03, RPRG94]). One of the projects, devoted to this problem is DIAsDEM[1].

The major aim of DIAsDEM is the incorporation of legacy data and semi–structured documents into an integrated information system. Let us review data sources, considered for integration by the DIAsDEM project.

Semi–structured documents relate to text databases. Text databases are data sources that contain word descriptions for objects. These word descriptions are usually not simple keywords but rather long sentences or paragraphs, such as product specifications, errors or bug reports, warning messages, summary reports, notes, or other documents. Text databases may be highly unstructured (such as some Web pages). Some text databases may be somewhat structured, that is, semi–structured (such as XML–documents), while other are relatively well structured (such as library databases).

Unstructured and semi–structured data does not possess a schema (in the conventional sense) and therefore, special methods should be applied to them in order to determine general descriptions of objects classes, as well as keyword or content associations, and the clustering behavior of text objects.

Knowledge discovery methods deliver results with a certain reliability, and obviously, the less structure is contained within the text sources the less information upon which

[1]Part of this work has been supported by the German Science Foundation DFG (grant no. CO 207/13–1); project DIAsDEM: Data Integration for Legacy Systems and Semi–Structured Documents Employing Data Mining Techniques.

the methods could rely, is provided and in its turn the worse reliability of the final result is delivered, and vice versa.

Many enterprises obtain *legacy databases* as a result of the long history of information technology development (including the application of different hardware and operating systems). A legacy database represents a system whose semantics is often not known.

A heterogeneous database consists of a set of interconnected, autonomous component databases. The components communicate in order to exchange information and answer queries. Objects in one component database may differ greatly from objects in other component databases, making it difficult to assimilate their semantics into the overall heterogeneous database.

Heterogeneity could arise due to two reasons. First of them is the difference between component databases. Component databases could use different data models and as a consequence, the same concepts could be differently modelled. This leads to a presence of different structures in schemas. Besides that, the nature of conflicts could differ from various integrity constraints to different query languages, etc. There exist quite powerful methods, capable of resolving such conflicts and therefore, such heterogeneity does not represent a serious problem for integration. On the contrary, the second reason giving rise to heterogeneity is caused by the absence of unified understanding about meaning and interpretation of the same data or the data that belongs together. This semantic heterogeneity presents a serious problem because there is no common case solution, but rather solutions for some specific problems.

Therefore, even if we do not consider semi–structured data sources with missing schema information, but only heterogeneous sources with known structure, in order to integrate such data, many conflicts have to be resolved to find correspondences between the given schemas and objects. There are also many methods for determining correspondences between the sources that as well as in the case of structure extraction, deliver results only with a certain degree of trustworthiness.

The integration of data from different sources into one information system is possible only when the (database) schema of each source is known and free of contradictions. Although available integration methods are capable of resolving some conflicts between the sources, they nevertheless presuppose precise knowledge about the structure within each source to be integrated and the correspondences between the sources.

Thus, to be able to integrate such sources as heterogeneous semi–structured data sources various methods determining their structure and correspondences can be applied, such as data mining, schema matching methods, etc. The uncertainty delivered by these methods cannot be taken into account by the integration methods since, as mentioned above, they presuppose precise information. Therefore, it is not possible to evaluate the integration result, i.e., to give a quantitative estimate of its quality.

We define our objective as developing such mechanisms that are capable of estimating the influence of uncertainties on the result of the integration process and, by that, to improve the integration quality.

1.2 Outline of the Thesis

In the previous section we have shown the necessity for developing a method enabling us to account for uncertainties and their influence on the integration result. In this thesis we create a framework for such a method, based on statistical analysis. The remainder of the thesis presents our approach and is organized as follows:

In Chapter 2 we review the integration process. Firstly, the integration of the sources with defined structure is presented. Secondly, the possible scenario of integration of heterogeneous data sources is given. Additionally we examine sources, that cause conflicts during the integration, and we provide a detailed classification of conflicts that occur during the integration of heterogeneous semi–structured data sources.

In Chapter 3 the methods that can be applied for solving specific problems in the integration process are reviewed. We consider the methods of knowledge discovery and data mining. They can be used to identify a structure for each source to be integrated. Then we examine the methods of schema matching, which are used for defining correspondences between data sources on schema level. Also, schema and data integration methods are analyzed in detail. The problem of data quality is then reviewed.

Chapter 4 provides us with a detailed analysis of the uncertainty. For that, we examine examples of integration of data sources both with and without uncertainties, in order to see on a contrast how they complicate estimation of the final result. In addition, we propose our classification of the uncertainties.

In Chapter 5 the idea and basic principles of the Latent Variable Model are introduced. Special attention is given to its part, the Latent Class Analysis (LCA) method, applicable for our task. LCA's theory and two possible solution scenarios are studied, the first is based on the system of linear equations and the second is on the maximum likelihood estimation method.

Chapter 6 is devoted to the questions of applying LCA to the integration task. For that we expand [AC03a, AC03b, AC03c] and build the statistical model of the integration task. Then we demonstrate how the integration process could be evaluated making use of the LCA on the examples. Besides that, we elaborate on the issues of improving the integration quality.

Chapter 7 concludes the thesis with a brief summary of the problem statement, the major contributions and the outlook.

2 Integration Process

In this chapter we review the issues related to the integration process. For that in Section 2.1 we, at first, describe the integration process for the data sources with defined structure, and then for those with undefined structure. Then in Section 2.2 we study the sources that trigger the conflicts during the integration, as well as the conflicts themselves. Section 2.3 summarizes this chapter.

2.1 Description of the Integration Process

Before considering the integration process for the heterogeneous semi–structured data or legacy systems we would like to consider the integration on the example of the sources with defined structure, i.e., for databases.

According to [SL90] *multidatabase systems* (MDBS) can be classified into two types based on the autonomy of the component *database systems* (DBSs): *nonfederated database systems* and *federated database systems* (FDBSs). A nonfederated database system is an integration of component *database management systems* (DBMSs) that are not autonomous. It has only one level of management and all operations are performed uniformly. In contrast to a federated database system, a nonfederated database system does not distinguish between local and nonlocal users. A particular type of nonfederated database system in which all databases are fully integrated to provide a single global schema can be called a *unified MDBS*.

A federated database system consists of component DBSs that are autonomous, yet participating in a federation to allow partial and controlled sharing of their data. Association autonomy implies that the component DBSs have control over the data they manage. They cooperate to allow different degrees of integration. There is no centralized control in a federated architecture because the component DBSs control access to their data. Thus, FDBS represents a compromise between no integration and total integration.

Depending on a degree to which the component DBSs are integrated, different architectures of FDBS can be used, for example, import/export–schema architecture ([HM85]), or multi–database architecture ([LMR90]), or the five–level schema architecture ([SL90]), etc. Besides that, various multidatabase system classifications exist in the literature: the taxonomy we used is given in [SL90], an alternative classification is, for example, proposed by authors in [BHP92]. In addition, integration can be carried out not on logical, but rather on the physical level, like for example, in case of *data warehouse*, which represents a repository of information collected from multiple sources/DBSs, stored under a unified schema, and which usually resides at a single site (for example, see [BG01, Leh03]). For our work, it is not important what exactly kind of integrated system with what type of architecture was chosen.

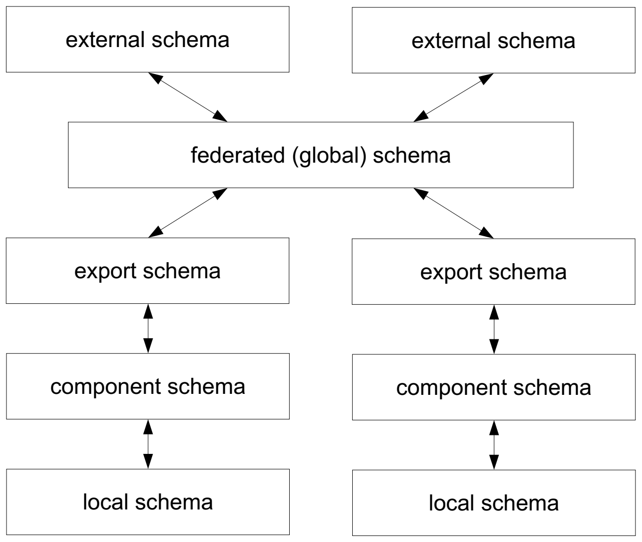

Figure 2.1: The five levels of the schema architecture

We consider the general case of the integration process on the example of the five–level schema architecture of an FDBS shown in Figure 2.1. This architecture includes the following schemas:

Local schema is the conceptual schema of a component DBS. A local schema is expressed in the local data model of the corresponding component system.

Component schema is derived by translating local schemas into a *canonical* or *common data model* (CDM). If a component DBS already uses the global data model as a local model, then local schema and component schema are the same.

Export schema represents a subset of a component schema that is available to the FDBS. If all data is to be exported, no separate export schema is needed.

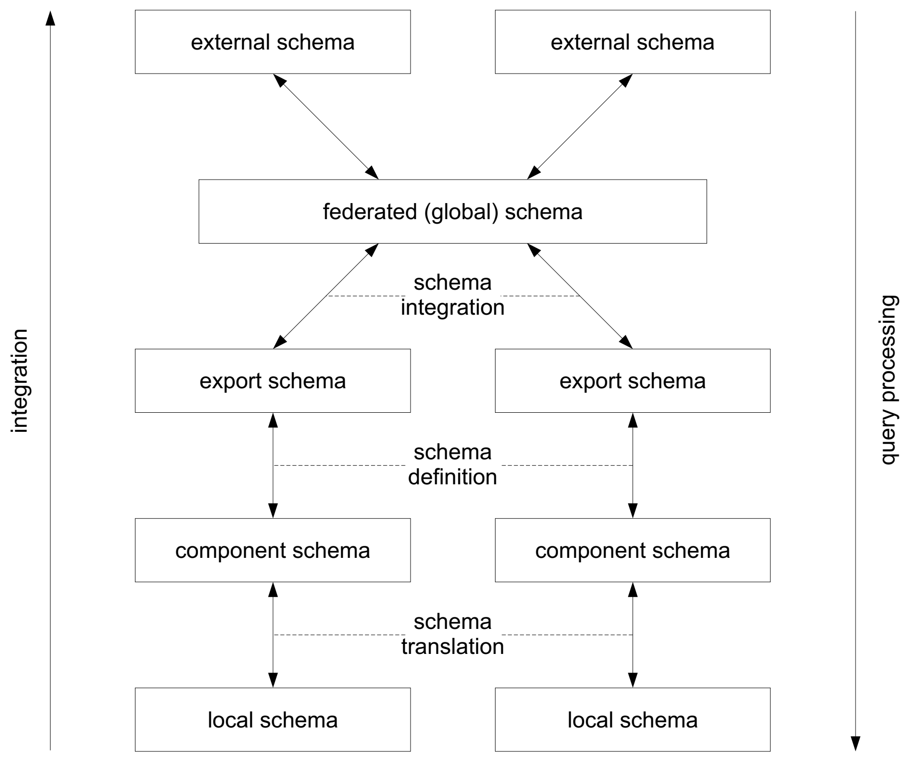

Figure 2.2: Bottom–up FDBS development process

Federated (global) schema provides an integrated view on all export schemas given for the component DBSs participating in a federation. The redundancy that could be found in the export schemas is removed in the federated schema. The structural differences and other conflicts between export schemas are resolved. The federated schema hides the distribution of data as well as all heterogeneities like different local data models and different ways of modelling or structuring data.

External schema defines a specific view on the federated schema for a user and/or application or a class of users and/or applications.

A bottom–up FDBS development process can be used to integrate several existing databases to develop an FDBS. Figure 2.2 illustrates the bottom–up process outlined below:

1. *Translate schemas.* Schema translation is performed when a schema represented in one data model (the source schema) is mapped to an equivalent schema represented in a different data model (the target schema). Schema translation is needed in two cases:

   ◇ Translating a local schema into a component schema when the DBMS's native data model is different from the CDM.

   ◇ Translating a part of the federated schema into an external schema when the external schema is expressed in a data model different than the CDM.

   In practice, the translation task may require more than just data model translation because the source and the target schemas may not be able to represent exactly the same semantics. Hence, schema translation poses two contradictory requirements: capture additional semantics during the schema translation that can later help in the task of schema integration, and maintain only the existing semantics because the local schema is not able to support the additional semantics. The translation should also be reversible in the sense that the component schema in the CDM should represent the same database represented by the local schema and it should be possible to translate a command on a component schema into command(s) on the corresponding local schema.

   Thus, schema translation is used for resolving heterogeneity conflicts caused by the presence of various data models, but in its turn can produce other schema level conflicts (for a possible conflict classification refer to Section 2.2).

2. *Define export schemas.* This step is performed by the administrators of respective component DBSs to authorize part of their databases to be included in the FDBS.

3. *Integrate schemas.* Schema integration refers to integrating selected export schemas into a single federated schema. This step includes analyzing and comparing the objects of the schemas to be integrated, including identification and resolving naming conflicts, domain conflicts, structural differences, constraint differences, missing data, etc. Besides that, the interrelationships among the schema objects should be specified at this step. The schema integration can be carried out right after resolving all schema level conflicts. At the same step data conflicts have to be resolved (see Section 2.2) for carrying out the logical data integration, which ensures correct query processing in/on the FDBS.

4. *Define external schemas.* If necessary, define external schemas for each federation user or class of federation users. If the data model of the external schema is different than the CDM then the schema should be translated.

We have considered a possible integration scenario. Obviously, even if the data structure is defined, the process of integration is not straightforward. It becomes more complicated because we aim at the incorporation of legacy data and heterogeneous non– or semi–structured data sources into an integrated information system.

Thus, as indicated above, in order to integrate data from different sources into one information system, the (database) schema of each source must be known and free of contradictions. Only then the integration process can start to remove structural and semantic conflicts between the sources. In the general case, neither legacy data nor semi–structured documents fulfill these prerequisites. Many collections of legacy data have no way of enforcing integrity constraints or of ensuring that constraints incorporated into application code are always enforced. For semi–structured data, not even a schema (in the conventional sense) is available, existing metadata is neither obligatory nor can be used for identifying identical objects.

We address these problems in order to enable an integration of legacy and semi–structured data following the traditional database integration techniques. In particular, we distinguish the following steps:

- ◇ Detecting dependencies and identifying semantics–carrying structure within each source.
- ◇ Finding the correspondences between the sources on the schema and instance (data) level.
- ◇ Applying results obtained on previous steps for schema and data integration.

According to these steps and to the integration process considered earlier for creating a FDBS we can present the integration process for legacy and semi–structured data with respect to the problems, which have to be resolved at each integration step. Such representation lets us abstract from type of architecture and integrated system, hence we are able to consider the general case. This integration process can be depicted as shown in Figure 2.3. Further we take a closer look at each of its steps.

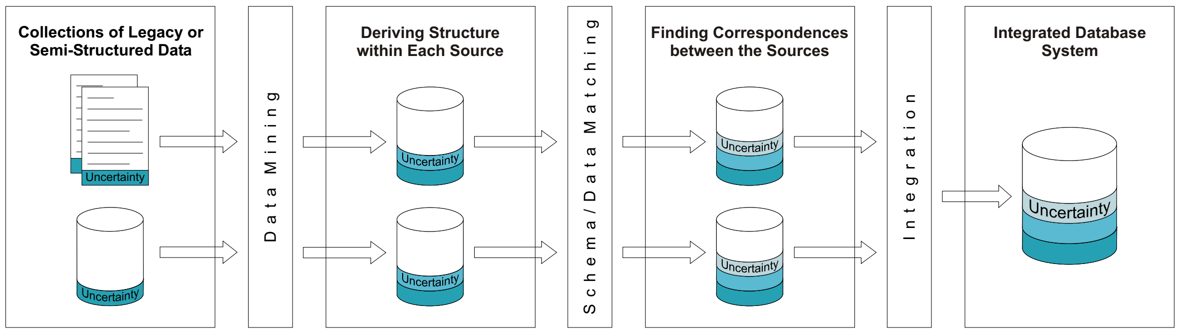
Collections of Legacy or Semi-Structured Data
Uncertainty
Uncertainty
Data Mining
Deriving Structure within Each Source
Uncertainty
Uncertainty
Schema/Data Matching
Finding Correspondences between the Sources
Uncertainty
Uncertainty
Integration
Integrated Database System
Uncertainty

Figure 2.3: The integration process

Detecting dependencies and identifying semantics–carrying structures within each source is a relevant task for the legacy systems and semi–structured sources due to the following reasons:

◇ Database schemas of component systems are often not complete, i.e., they do not represent the complete semantics of the application.

◇ The documents containing data are usually semi–structured or a database schema is not available.

◇ In legacy systems the semantics of database schemas is not completely known.

◇ Correspondences and dependencies among data within each source are also not (completely) known. Sometimes even experts might only guess, which correspondences are valid.

Thus, the way it is schematically shown in Figure 2.3, data sources, which we would like to integrate, already contain uncertain information. The presence of the uncertainties hinders the integration, thus at this step one should solve the problems listed above, which relate to the structure discovery problem that is the problem of finding and characterizing the underlying structure of a data set. Depending on the specific application requirements, different techniques can be used to discover a good data representation. One can use data mining techniques that refer to a class of technologies designed to assist in the discovery of knowledge in large data sets. Data mining is an essential process where intelligent methods are applied in order to extract data patterns. We describe them in the next chapter.

Here we view the steps that are to be followed in order to detect dependencies and find structure within each source.

◇ *Choosing data mining techniques.* In this step the existing mining techniques should be checked with respect to their applicability to our task. One should include into consideration such primitives as, for example, the kinds of knowledge to be mined, interestingness measures for pattern evaluation, etc.

◇ *Mining the data.* This step corresponds to classical data analysis. Here, for example, classification methods could be applied to generate and then check hypotheses on dependencies within legacy data. As well, hypotheses of the application expert can be checked and possibly refined. The detection of relationships among values in legacy data can provide additional indications on the existence of further dependencies. For that, for example, dependency modelling techniques can be applied. For analyzing semi–structured data the methods for pattern recognition can be used. The aim here is to detect structural properties which indicate semantic properties. Thereby, pieces (fragments) of a conventional schema should be found.

- ◇ *Analysis of mining results.* The results obtained from the data mining techniques have to be elaborated and generalized. Besides that, it should be taken into account, that these results always hold certain probabilities. The uncertainties originate from both the data sources, which contain uncertain information and the process of knowledge discovery. In a case, where not all data and/or data states are available, none of the methods of data mining ever extract the full and complete information from data in databases. In realistic scenarios we typically have only a restricted collection of data (representing a single or very few database states), such that incompleteness and uncertainty is always present.

Finding the correspondences between the sources on the schema and instance (data) level. After the structure and the correspondences in each source are found, the next integration step can be carried out, namely, comparison of the input schemas, which is necessary for:

- ◇ Identifying inter–schema relationships
- ◇ Detecting and resolving possible schema conflicts

In order to resolve these problems, the schema matching methods, that we consider in detail in the next chapter, could be employed. In the following, shortly listed are the steps that must be undertaken in order to apply schema matching for finding correspondences between schemas:

- ◇ *Choosing schema matching techniques.* In this step the matching techniques should be chosen. The choice is based on many parameters, for example, such as: schema language (relational, XML schemas, etc.), the kind of input data that can be used (schema information, data instances, dictionaries etc.), etc.
- ◇ *Schema Matching.* In this step a number of schema matching methods could be applied in various combinations.

  In case when useful schema information is limited, as is often the case for semi–structured data, a schema can be constructed from instance data. The use of instance–level matchers can be valuable to detect incorrect interpretations of schema information. For example, constraint– or linguistic–based methods can be used for extracting probabilistic structures of data objects. Schema–level matchers are reasonable to apply when we have knowledge about structure, but not sure about the assignment of data objects to classes or about the extensional correspondence between classes from two data sources. The constraint– and linguistic–based matchers mentioned above can as well be used as schema–level matchers. For example, linguistic–based matchers that use names and text (i.e., words or sentences) can be applied to find semantically similar schema elements. Schemas often contain constraints to define data types and value ranges, relationship types and cardinalities, etc. If both input schemas contain such information, it can be used by a constraint–based matcher to determine the similarity of schema elements.

◇ *Analysis of matching results.* The output of a match system is a mapping indicating which elements of the input schemas correspond to each other, i.e., match. Basically matching prototypes determine correspondences between schema elements and use similarity values between 0 (strong dissimilarity) and 1 (strong similarity) to indicate the plausibility of the correspondences. Besides that, various quality measures enable the comparison of the output quality of different matching systems. Obviously, any schema matching method adds uncertainty into the integration process.

After the correspondences between the schemas are determined, the instance level conflicts must be resolved and the correspondences between the specific data records must be determined. In order to do that, numerous data cleaning methods can be applied. We distinguish the following steps:

◇ *Choosing data cleaning techniques.* In this step the data cleaning techniques should be chosen that are appropriate to our task. One should distinguish between data cleaning methods intended for resolving the single–source problems that refer to errors and inconsistencies in the actual data, which are not visible at the schema level and the multi–source problems that refer to identifying matching records.

◇ *Data Cleaning.* Depending on the kind of the conflicts that must be resolved and on the additional information at one's disposal, we can employ various methods of data cleaning. Data cleaning methods can be divided into *inference–based*, where cleaning is performed by discovering patterns within the data and using those patterns to derive rules for data cleaning and *data–based*, usually applied to clean a specific type of data such as name and address data where cleaning rules are pre–specified in the cleaning tool and cleaning is performed by finding data that matches those rules and mapping to the cleaned value by using lookup tables. We discuss data cleaning methods in detail in the next chapter.

◇ *Analysis of data cleaning results.* Obviously, that as well as the rest of the methods being used in the integration process, the methods of data cleaning do not deliver precise results, but rather results with certain reliability, in this way increasing the uncertainty of the integration process. That is why, at this step the results of the data cleaning methods must be evaluated, for which one could employ various data quality metrics such as, for example, accuracy, consistency, completeness, etc.

Applying results obtained on previous steps for schema and data integration. Let us first consider the schema integration process that consists of the following steps:

- ◇ *Choosing schema integration technique.* Since not all the conflicts at the schema level could have been resolved by the previous methods, one should choose such a schema integration method that would be capable of resolving the rest of the conflicts. Aside from that during the selection of the schema integration technique, it is important to take into account the kind of modelling concept supported by the schema integration method. During the selection of the integration method, the criteria of the integration quality should be defined, based on which one would be able to judge about the integrated schema quality.
- ◇ *Schema integration.* At this step the designer has to decide what kind of the strategy will be used to integrate the given schemas. Answers to the following questions should be found: do the schemas to be integrated have different weights of relevance for the integration? In what order should the schemas be integrated? Should they be integrated simultaneously? Further for conducting the integration, the additional information gathered at the preceding stages could be needed. For example, if we need to integrate a relational schema with an object–oriented schema and we consider, that both of them have the same weights of relevance for the integration, then in order to account for the information received from the schema matching about the correspondences between sources for integration, it is advisable to use the Model Independent Assertions method (see Section 3.3). This method is capable of resolving some structural conflicts that could exist between schemas and fully resolves all of the heterogeneity conflicts. In the next chapter the integration methods are considered in detail.
- ◇ *Analysis of schema integration results.* The integrated schema has to fulfill different quality criteria, for example, minimality, correctness, etc. Here it is important to note that these criteria judge about the capabilities of the integration technique and during the calculation take into account neither data quality, nor the information about the data reliability, received from the methods at the preceding steps.

Data integration concludes the process of integration. For data integration the data cleaning methods described above can be used. Here one should consider what type of data integration was chosen: physical or logical, at which in contrast to physical integration, data must be only correctly brought together in case of queries.

Thus, the above described integration process could be presented as a chain of the successively applied methods, aimed at the resolving certain conflicts. In the next section we analyze possible reasons triggering these conflicts and then give detailed classification of the conflicts themselves.

2.2 Schema and Data Integration Conflicts

Since a database is defined by its schema and data, we classify conflicts at the highest level as either schema or data conflicts. Schema conflicts result from the use of different schema definitions in different databases. Data conflicts are due to inconsistent data in the absence of schema conflicts.

First we consider schema conflicts. There are three basic causes of schema conflicts.

- ◇ *Different modelling perspectives.* The different viewpoints that groups of users and designers have about certain information during the design phase.
- ◇ *Different modelling constructs.* The existence of different kinds of constructs in the data models, offering different modelling possibilities.
- ◇ *Incompatible design specifications.* A different choice for the common parts of the various schemas regarding names, types, integrity constraints, etc. results in different schemas.

According to earlier research [BLN86, SCS99, SPD92] four types of conflicts occur:

- ◇ Semantic conflicts
- ◇ Descriptive conflicts
- ◇ Heterogeneity conflicts
- ◇ Structural conflicts

Semantic conflicts. Considering two independently designed schemas describing semantically intersected universes of discourse, one could find mutually corresponding classes (or relations), whose objects, however, do not represent fully identical sets of the real–world objects. Thus, we can distinguish four cases: similar objects can be presented as equivalent, overlapping, included or disjoint sets.

Descriptive conflicts. This type of conflict occurs, when the same data item is represented differently in two schemas. Descriptive are such conflicts as: *name conflicts*, when the schemas define names for classes and attributes in different ways (homonyms and synonyms), *attribute conflicts*, when the semantically equivalent attributes describe a property with different precision or homonyms and synonyms occur on value level, *integrity constraints conflicts*, when the set of object states or attribute values for corresponding classes or attributes are restricted differently.

Heterogeneity conflicts. They occur when different data models are used for the schemas, for example, a relational one and an object–oriented one. This conflict type is especially relevant when such schemas have to be integrated. Then one should decide which data model should be preferred for the integrated schema. Hereby, it should be noted that the modelling concept of a relational schema is not as rich as that of an object–oriented. Also, for modelling of the same things different data models, as a rule, use different modelling concepts. As a consequence, the heterogeneity conflicts often implicitly include the next type of conflicts, namely, structural conflicts.

Structural conflicts. Since the way in which designer models a real–world aspect is not unique, even if only one data model is used, the structural conflicts appear. In other words, the structural conflicts occur when the same real–world object described in two schemas use different elements of the same data model. The most frequent type of the structural conflict appears between an attribute and a class, when an attribute of a class of one schema corresponds to a class of the other schema.

The authors [KS91, KCGS] suggest an alternative classification for the above described conflicts based on the relational model. And then they extend it for the object–oriented model. Below we describe a conflict classification for the relational model.

As an outcome of the fact that the relational model uses either tables or attributes to represent information, we can classify schema conflicts within this model completely by enumerating combinations of different structures used to represent information and all possible specifications of the structures. Then the classification looks like as follows:

1. Table–versus–table conflicts
   - A. One–to–one table conflicts
     - a. Table name conflicts
       - ◇ Different names for equivalent tables
       - ◇ Same name for different tables
     - b. Table structure conflicts

            - ◇ Missing attributes
            - ◇ Missing but implicit attributes
        - c. Table constraint conflicts
    - B. Many–to–many table conflicts
2. Attribute–versus–attribute conflicts
    - A. One–to–one attribute conflicts
        - a. Attribute name conflicts
            - ◇ Different names for equivalent attributes
            - ◇ Same name for different attributes
        - b. Default value conflicts
        - c. Attribute constraint conflicts
            - ◇ Data type conflicts
            - ◇ Attribute integrity–constraint conflicts
    - B. Many–to–many attribute conflicts
3. Table–versus–attribute conflicts

Table–versus–table conflicts occur when different databases use different definitions to represent information in tables (for example, different names, structures, or constraints on the tables). Table–versus–table conflicts can be decomposed into one–to–one table conflicts and many–to–many table conflicts.

One–to–one table conflicts can occur when databases represent the same information in single tables using different names, structures, and constraints. Thus, one–to–one table conflicts are further decomposed into table name conflicts, table structure conflicts, and table constraint conflicts.

Table name conflicts arise due to different names assigned to tables in different databases. There are two types: conflicts due to the use of different names for semantically equivalent tables and conflicts due to the use of the same name for semantically different tables.

Table structure conflicts arise from differences in the number of attributes in database tables, that is, when a table in one database is missing some attributes in a corresponding table in another database. There are two interpretations for missing attributes: the attributes are indeed missing, or the missing attributes are implicit and can be deduced.

Table constraint conflicts arise from differences in the specifications of tables constraints. For the database language SQL this type of conflict includes four subcategories:

candidate key definition, primary key definition, foreign key definition, and check condition.

Many–to–many table conflicts occur when databases use different numbers of tables to represent the same information. This type of conflict may frequently occur since database users may define tables in different ways.

Some conflicts may combine many–to–many table conflicts with subcategories of one–to–one table conflicts. However, separate categories are not required for such compound conflicts because they can be decomposed into basic conflicts.

Attribute–versus–attribute conflicts are caused by different definitions for semantically equivalent attributes in different databases, including different names, attribute data types, and integrity constraints. Like the table–versus–table conflicts, these conflicts can be decomposed into one–to–one and many–to–many conflicts.

One–to–one attribute conflicts arise due to different definitions for semantically equivalent attributes in different tables. The attribute definition consists of the attribute name, data type, constraints, and default values. Where the data type specification for an attribute can be considered as a special case of its constraint definition.

Attribute name conflicts are similar to the table name conflicts discussed earlier. There are two types: one arising from the use of different names for semantically equivalent attributes in different databases and the other arising from the use of the same name for semantically different attributes.

Default value conflicts arise when the default values of semantically equivalent attributes in different databases are different.

Attribute constraint conflicts are further decomposed into data type and attribute integrity–constraint conflicts, which occur when semantically equivalent attributes in different databases have different data types or respectively different attribute integrity–constraints.

Many–to–many attribute conflicts. As remarked for many–to–many table conflicts, these conflicts may combine many–to–many attribute conflicts with subcategories of one–to–one attribute conflicts. Again, there is no need for separate categories for such compound conflicts because they can be decomposed into several types of basic conflicts.

Table–versus–attribute conflicts occur if some databases use tables and others use attributes to represent the same information. This conflict can be regarded as a combination of many–to–many table conflicts and many–to–many attribute conflicts.

Since the above classification results from the combinations of different structures and all possible specifications of the structures, the number of the conflicts represented in the classification will grow if the data model is richer than the reviewed relational model.

That is because the richer the data model, the more different combinations, which cause conflicts, exist. In this way the classification becomes non transparent. Moreover, this classification hardly adds new features to the classification given in the beginning of this section, which supposes four types of conflicts: semantic, descriptive, heterogeneity, and structural. For our purposes it is sufficient to use this four–conflict–types classification.

Thus far, our discussion has been focused on the classification of the schema conflicts. Below we examine the four types of data conflicts:

1. Incomplete data (missing values)
2. Wrong data
    A. Incorrect–entry data
    B. Obsolete data
3. Noisy Data
4. Different representation for the same data (same representation for different data)
    A. Different expressions
    B. Different units
    C. Different precisions

Incomplete data means, that some tuples can have no recorded value for several attributes.

Wrong data generally arises due to failures in maintaining a database, such as inability to keep the database up to date and to enforce integrity constraints. We can distinguish two types of conflicts: *incorrect–entry data*, when equivalent attributes in different databases, which are expected to have the same value, have different values and *obsolete data*, when the equivalent attributes have different values because one of the values was not updated in time.

Noisy data is the data containing errors or outlier values that deviate from the expected. Noise is defined as a random error or variance in a measured variable.

Different representation for the same data. There are three aspects to the representation of data: expressions, units, and precisions.

Different expressions. Conflicts of this type arise when different scalar values are used to represent the same data. Of particular interest are cases when different expressions describe the same piece of information and when different databases use separate codes to denote the same data.

Different units. Conflicts of this type arise when numerical data denoting the same physical quantity are represented in different units across databases. Different units give differing meanings to numeric data.

Different precisions. This type of conflict arises when semantically equivalent attributes have values from domains with different cardinalities. This difference in cardinality results in different scales of precision for similar data.

In practice schema and data conflicts usually emerge not alone, but rather in different combinations. Discrepancies between schemas as well as between data usually show a mix of conflict types and should be solved together.

2.3 Summary

In Section 2.1 we have given a possible multidatabase system classification. Then, using data sources with defined structure as an example, a possible scenario of integration process for creating a federated database system with use of the five–level schema architecture, has been reviewed. Such a process in general consists of four steps: schema translation, schema definition, schema integration and, if necessary, external schema definition. At that, at each step the specific problems must be resolved, which makes the integration process complicated.

Since our primary task is integrating the data sources with undefined structure, such as legacy data or heterogeneous semi–structured data, integration process becomes even more complicated. In order to review the general case of integration of such sources independently of the specific architecture or integration system, we proposed to consider integration process, subject to methods that have been used during the integration.

At the first step, data mining methods could be used in order to find the structure within each source. At the next step, methods of schema matching could be employed in order to determine correspondences between the sources at the schema level, whereas data cleaning methods could be used in order to find corresponding data records. After the correspondences in each source and across sources are found, methods of integration could be used to carry out integration itself.

Section 2.2 considers the conflicts that need to be resolved by the methods mentioned above and the causes of these conflicts. One should distinguish the schema level and the data level conflicts. According to the classification that is most commonly used in the literature, schema conflicts could be divided into semantic, descriptive, heterogeneity and structural conflicts. There also exist more comprehensive classifications, in accordance with the enumeration of all possible structure combinations that are used in order to present information in schema, for example, different names for equivalent tables, same name for different attributes, etc. Data conflicts could be subdivided into incomplete, wrong, noisy data. Besides that, there are such conflicts that the same data has different representation or different data has the same representation.

It is important that regardless of the fact that each of the methods used during the integration resolves only certain types of conflicts, helping in this way (and in some cases making integration possible), any of those methods output result with certain trustworthiness, thus introducing additional uncertainty into the integration process.

In the next chapter we concentrate on the methods described in this chapter.

3 Related Work

In the previous chapter the integration process has been reviewed and its essential stages have been distinguished. In this chapter we consider the questions related to the specific problems intrinsic to each of those stages along with the possible methods for their solution. The chapter is organized as follows: in Section 3.1 we touch the problem of knowledge discovery and data mining, which is concerned with the task of information discovery from the data in databases. Section 3.2 refers to the second step of the integration process and is concerned with questions of schema matching, which searches for corresponding elements in different schemas. Sections 3.3 and 3.4 elaborate on the methods of the schema and respectively data integration. Section 3.5 reviews the problem of data quality. The final Section 3.6 concludes the chapter providing a summary of the important related work results.

3.1 Knowledge Discovery in Databases and Data Mining

Data Mining and Knowledge Discovery in Databases (KDD) have become important subjects of research in the recent years and a vast amount of publications on these topics are now available [CHY96, FPSS96, HC98, HK01, HMS01, PS96].

KDD means a process of nontrivial extraction of implicit, previously unknown, and potentially useful information from data in databases. The term process implies that KDD consists of many steps, which involve data preparation, search for patterns (extracting structure from data), knowledge evaluation, and refinement, all repeated in successive iterations.

Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns (or models) over the data.

The knowledge discovery goals are defined by the intended use of the system. We can distinguish two types of goals: *verification* and *discovery*. With verification, the system is limited to verifying the user's hypothesis. With discovery, the system autonomously finds new patterns. The scope of our work concentrates only on the latter. We further subdivide the discovery goal into *prediction*, where the system finds patterns for predicting the future behavior of some entities, and *description*, where the system finds patterns for presentation to a user in a human–understandable form. The classification of the data mining methods with respect to their goals is presented below. It is necessary to note, that the boundaries between prediction and description are not sharp, some of the predictive models can be descriptive and vice versa.

A. Prediction

- ◇ *Classification* is the process, which finds the common properties among a set of data items (objects) in a database and classifies them into different classes, according to a classification model.
- ◇ *Regression* is learning a function that maps a data item to a real–valued prediction variable.

Prediction can be viewed as the construction and use of a model to assess the class of an unlabeled sample, or to assess the value or value ranges of an attribute that a given sample is likely to have. In this view, classification and regression are two major types of prediction problems, where classification is used to predict discrete or nominal values, while regression is used to predict continuous or ordered values.

B. Description

- ◇ *Clustering* is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data. The categories can be mutually exclusive and exhaustive or consist of a richer representation, such as hierarchical or overlapping categories.
- ◇ *Summarization.* Data and objects in databases often contain detailed information at primitive concept levels. Data Summarization is a process, which abstracts a large set of relevant data in a database from a low concept level to relatively high, in other words, it involves methods for finding a compact description for a subset of data.
- ◇ *Dependency modelling* consists of finding a model that describes significant dependencies between variables. Dependency models exist at two levels: the structural level of the model specifies (often in graphic form) which variables are locally dependent on each other and the quantitative level of the model specifies the strengths of the dependencies using some numeric scale.

Nowadays the array of different algorithms is enormous, but they all tend to differ primarily in the goodness of fit criterion used to evaluate model fit or in the search method used to find a good fit and most methods can be viewed as extensions or hybrids of a few basic techniques and principles listed above.

As mentioned above data mining derives patterns within the data. By its nature such a process cannot yield precise results, because the mining of the data is not a straightforward procedure, multiple data conflicts have to be resolved, and the implicit semantics of the data should be interpreted. This presents the most difficult part in the whole process, because even experts are often only able to guess the semantics. As a consequence, data mining results are never unambiguous.

This way a data mining process may generate a large number of patterns, although specification of the task–relevant data and of the kind of knowledge to be mined may substantially reduce the number of patterns generated. Typically, only a small fraction of these patterns will actually be of interest to the given user. Thus, users need to further confine the number of uninteresting patterns returned by the process. This can be achieved by specifying quality measures that estimate the *simplicity*, *certainty*, *utility*, and *novelty* of patterns. These objective measures are based on the structure of patterns and the statistics underlying them. In general, each measure is associated with a *threshold* that can be controlled by the user. Rules that do not meet the threshold are considered uninteresting and hence are not presented to the user as knowledge.

Simplicity. A factor contributing to the quality of a pattern is the pattern's overall simplicity for human comprehension. Objective measures of the pattern simplicity can be viewed as functions of the pattern structure, defined in terms of the pattern size in bits, or the number of attributes or operators appearing in the pattern.

Certainty. Each discovered pattern should have a measure of certainty associated with it that assesses the validity or "trustworthiness" of the pattern. A certainty could be measured, for example, by a *confidence*, which denotes the conditional probability of the fact that a pattern was found correctly, or *precision*, which refers to a proportion of patterns retrieved that are relevant, or *recall*, which refers to a proportion of target patterns that are retrieved.

Utility. The potential usefulness of a pattern is a factor defining its quality. It can be estimated by a utility function, such as *support*. The support of a pattern refers to the percentage of task–relevant data tuples for which the pattern is true.

Novelty. Novel patterns are those that contribute new information or increased performance to the given pattern set. For example, data exception may be considered novel in that it differs from that expected based on a statistical model or user beliefs. Another strategy for detecting novelty is to remove redundant patterns.

Data mining systems allow users to flexibly and interactively specify, test, and modify quality measures and their respective thresholds. There are many other objective measures, apart from the basic ones studied above. Subjective measures exist as well, which consider user beliefs regarding relationships in the data, in addition to the objective statistical measures.

In the context of the DIAsDEM project a method utilizing iterative clustering algorithm to derive structured XML DTDs (Document Type Definitions) in order to extend previously derived flat DTDs was developed [GSW02, GWS01, WS02].

This method pursues two objectives for a given archive of text documents: all text documents should be semantically tagged and an appropriate, preliminary flat XML DTD should be derived for the archive. Semantic tagging is a two–phase process. At the first phase knowledge discovery methods are employed in order to build clusters of semantically similar text units, to tag documents in XML according to the results and to derive an XML DTD describing the archive. The knowledge discovery process results in a final set of clusters whose labels serve as XML tags and DTD elements. Huge amounts of new documents can be converted into XML documents in the second, batch–oriented and productive phase of the framework. All text units contained in new documents are clustered by the previously built text unit clusters and are subsequently tagged with the corresponding cluster labels.

Since the semantic annotations are derived with data mining techniques, they have some level of confidence. Thus, it is essential that the validity of each tag is expressed in quantitative terms and is estimated properly. Furthermore, an ordering should be imposed upon the tags. Hence, after deriving semantic XML tags, they are combined into a probabilistic DTD by deriving the most likely ordering of the tags and computing the statistical properties of each tag inside the document type definition.

3.2 Schema Matching

A schema of any source consists of a set of related elements, such as tables, columns, classes, or attributes, or XML elements. Matching is a schema manipulation operation, which takes two schemas as an input and returns a mapping that identifies corresponding elements in the two schemas. Each mapping element can have a mapping expression, which specifies how these elements are related.

The criteria used to match elements of two schemas are based on heuristics that are not easily captured in a precise mathematical way. Thus, schema matching is inherently subjective. Schemas may not completely capture the semantics of the data that they describe, and there may be several plausible mappings between two schemas (making the concept of a single best mapping ill–defined). This subjectivity makes it valuable to have user input to guide the match and essential to have user validation of the result. This guidance may come via an initial mapping, a dictionary or thesaurus, a library of known mappings, etc. Thus, the goal of schema matching is: given two input schemas in any data model and, optionally, auxiliary information and an input–mapping, compute a mapping between schema elements of the two input schemas that passes user validation.

Various methods of schema matching have been developed (e.g., MOMIS [BBGV01], LSD [DDL00a, DDL00b], DIKE [PTU00], Clio [MHH00, YMHF01]). Schema matchers can be characterized by the following criteria (a detailed survey based on this taxonomy can be found in [RB01]):

- ◇ Schema–level matchers
- ◇ Instance–level matchers
- ◇ Element–versus–Structure granularity
- ◇ Constraint–based matchers
- ◇ Linguistic–based matchers
- ◇ Auxiliary information
- ◇ Matching cardinality
- ◇ Combining different matchers

Schema–level matchers. These matchers consider only schema information, not instance data. Schema information contains knowledge about schema structure, constraints, data types, description, name, different kinds of relationships, etc.

Instance–level matchers. Matching approaches consider instance data. They either use metadata and statistics derived from data instances to annotate the schema or directly find related schema elements.

Element–versus–Structure granularity. A structure–level matching computes combinations of elements that appear together in a schema. An element–level matching computes a mapping between individual schema elements.

Constraint–based matchers. These matchers use schema constraints to determine the similarity of schema elements. For example, similarity can be based on the equivalence of data types and domains, of key characteristics, of relationships cardinality, etc.

Linguistic–based matchers. This type of matchers uses names and other textual descriptions to find semantically similar schema elements.

Auxiliary information. Schema matchers may rely on auxiliary information, such as dictionaries, global schemas, previous matching decisions, and input match–mismatch information.

Matching cardinality. Schema matchers may compute mappings between elements of two schemas with different cardinality, namely 1:1 and the set–oriented cases 1:n, n:1, and n:m.

Combining different matchers. An individual matcher uses a single algorithm to perform the match. A more effective way is to run independent match algorithms on the two schemas and then combine the results (*Multiple matchers*), or use multiple criteria to perform the matching (*Hybrid matchers*).

Another important aspect is the evaluation of the matching quality because, as mentioned before, schema matching is a subjective process and as well as in case of data mining, it cannot provide a user with precise results.

According to [DMR03] the match task first has to be manually solved in order to provide a basis for evaluating the quality of automatic matching strategies. The obtained real match result can be used as the standard to assess the quality of the result automatically determined by the match system. Comparing the automatically derived matches with the real matches results in the four sets that can be used to define quality measures for schema matching. The set of derived matches is comprised of the *true positives* and the *false positives. False negatives* are matches needed, but not automatically identified, while false positives are matches falsely proposed by the automatic match operation. *True negatives* are false matches, which have also been correctly discarded by the automatic match operation. Intuitively, both false negatives and false positives reduce the match quality.

Based on the cardinality of these sets, two common measures, *precision* and *recall* can be computed, where precision describes the correctness of the matching result. It is defined as the share of the true positives among all derived matches. Recall describes the completeness of the matching result. It is defined as the share of the true positives among all real matches. In the ideal case, when no false negatives and false positives are returned, precision and recall are equal to 1. However, neither precision nor recall alone can accurately assess the match quality. In particular, recall can easily be maximized at the expense of a poor precision by returning all possible correspondences, i.e., the cross product of two input schemas. On the other side, a high precision can be achieved at the expense of a poor recall by returning only few (correct) correspondences. Hence, it is necessary to consider both measures or a combined measure. *Overall* is an example of a combined measure, which was developed specifically in the schema matching context and embodies the idea to quantify the post–match effort needed for adding false negatives and removing false positives.

The next aspect that we should be aware of is that despite the large variety of available methods of schema matching, none of them can consider uncertain input information.

One of the examples of combining different matchers is the Cupid approach [MBR01]. Cupid is schema–based and includes automated linguistic–based matching and is both element–based and structure–based. It is biased toward similarity of atomic elements (i.e., leaves), where a large part of schema semantics is captured.

Cupid computes similarity coefficients between elements of the two schemas and then deduces a mapping from those coefficients. The coefficients are calculated in two phases. The first phase, called linguistic matching, matches individual schema elements based on their names, data types, domains, etc. A thesaurus is used to help match names by identifying acronyms and synonyms. The result is a linguistic similarity coefficient between each pair of elements. The second phase is the structural matching of schema elements based on the similarity of their contexts or vicinities. The structural match depends in part on the linguistic matches calculated in phase one. The result is a structural similarity coefficient for each pair of elements. The weighted similarity is a mean of linguistic similarity and structural similarity. A mapping is created by choosing pairs of schema elements with maximal weighted similarity.

3.3 Schema Integration

In the previous section we have reviewed possible methods for finding mapping between schema elements. After the needed set of matching elements has been found, the schema integration can be carried out. According to [Con97] we can distinguish four basic integration principles:

- ◇ Model Independent Assertions
- ◇ Upward Inheritance
- ◇ Formalized Object–Oriented Integration
- ◇ Generic Integration Model

Model Independent Assertions. This method assumes two phase integration process described in [SP94, SPD92].

First, similarities and discrepancies among input schemas have to be determined. This is the *investigation phase*. The basic idea is to examine input schemas to define the applicable set of inter–schema correspondences, which represents a problem of schema matching, described in the previous section. That is why the authors assume that all correspondences are defined and only consider second phase, that is *integration*. The integrated schema is built semi–automatically, according to the inter–schema correspondences and available integration rules.

This approach is based on a generic data model that represents a set of modelling concepts and allows to reason about integration of conflicting schemas. Thus, the method solves all of the heterogeneity conflicts and performs integration without transformation of initial schemas. The method supports heterogeneity of input schemas through a data–model–independent description of inter–schema correspondences and generic integration rules. For instance, a relational schema may be directly compared with an object–oriented schema.

This integration method allows to integrate not only schema elements, but also paths and links between elements. It automatically resolves some of the structural conflicts, for example, it makes it possible to integrate an entity type and a relationship type in the Entity–Relationship Model. That is due to the fact that in one schema a relation between objects could be presented as a relationship whereas in another schema the same relation could be modelled as an entity.

Semantic conflicts cannot be resolved using this method. That is because the method considers only semantically equivalent integration assertions. The method should be extended for integration of inclusion, intersection, and exclusion assertions.

This method supports neither a concept of generalization and specialization nor integration cases in which one object in one schema corresponds to a set of objects in the other schema. It also cannot integrate complex attributes and none of the descriptive conflicts could be resolved.

Upward Inheritance. Integration methodologies proposed in [SN88, BFN94, GSSC95] resolve semantic overlapping by introducing generalized classes (upward inheritance). That is, the original inheritance hierarchies are subhierarchies of the resulting integrated hierarchy. In other words, it is assumed that for each class of the local schemas there always exists a semantically equivalent class in the integrated schema and for all class–subclass–relations in the local schemas there exist same relations in the integrated schema. Thus, two classes can be integrated into one class if and only if they are semantically equivalent. If two classes are not semantically equivalent, but have overlapping extensions (sets of objects belonging to the classes), then for each class an independent class in the integrated schema should be created. If one class to be integrated is a subclass of another class to be integrated, then this class–subclass–relation should also be represented in the integrated schema. For two classes one superclass in the integrated schema should be created if they semantically overlap. Thus, this method allows to solve semantic conflicts.

Heterogeneity conflicts should be resolved since the method assumes that the schemas to be integrated are object–oriented. This always requires transformation into the object–oriented data model. Neither descriptive nor structural conflicts could be resolved using this method.

Formalized Object–Oriented Integration. In [RPG95, RPRG94] a methodology has been proposed for the creation of an integrated schema from a given set of local database schemas. This methodology involves acquisition of semantic knowledge pertinent to the objects of a local objects schema. During this knowledge acquisition process, for each property of a local object, parameters that contribute to the semantic meaning of the property are identified (such as meta–properties) and their values (meta–values) are captured. Further, concepts such as object equivalence class and property equivalence class are utilized to facilitate the creation of the integrated schema. Thus, the method partially solves semantic conflicts.

Similar to the upward inheritance method, this method supposes transformation into object–oriented data model and in this way resolves heterogeneity conflicts. The method does not resolve structural conflicts. Descriptive conflicts could be partially resolved making use of meta–properties and meta–values information.

Generic Integration Model. The integration strategy discussed in [SS96a, SS96b, CHJ$^+$96, Sch95] is based on the Generic Integration Model (GIM), which is a semantically poor data model and used as canonical data model. The advantage of using a semantically poor data model instead of a semantically rich data model is that a semantically rich data model causes more heterogeneity on the schema level. This heterogeneity on schema level in its turn increases the complexity of the integration process. This method is data–model–independent, i.e., this method is not concerned with the kind of data models the schemas have, regardless of whether these are the schemas to be integrated or the integrated schema. Thus, GIM solves most of the heterogeneity conflicts. In GIM only schema information, which is necessary for the integration process can be expressed by means of GIM concepts whereas schema information defining a specific view of an application cannot be expressed.

Besides that, the hierarchies which are to be found in the schemas for integration do not necessarily have to be in the integrated schema, thus this approach allows restructuring the initial hierarchies and is flexible in working with generalization and specialization that leads to a state, where the number of superclasses (among which are also the ones not needed by the integrated schema user) does not increase in the integrated schema in the way that it happens in the upward inheritance method.

GIM requires non–overlapping class extensions. The extensions of any two classes have to be either disjoint or identical. In case they overlap, the GIM method uses extensional decomposition. Then three disjoint extensions appear: one for the objects common to both classes, and other two, containing only the respective objects of each of the individual classes. Through the disjoint decomposition semantic conflicts are resolved. Descriptive and structural conflicts cannot be resolved using this method.

Speaking about the schema integration methods, it is necessary to touch upon the subject of their quality. Following [BLN86] there are four major quality criteria for schema integration:

◇ *Completeness.* The integrated schema must be a representation of the union of the application domains associated with the schemas. This means that there must not be a loss of information contained in local schemas. In general all methods are not complete. That is because the modelling concepts could always be found, such that, they are not supported by the given, specific integration methods. In some cases one can speak about the limited completeness, when in the integrated schemas only the modelling concepts supported by the specific methods are used.

◇ *Correctness.* The integrated schema must contain all concepts present in any component schema correctly. This means that for each element in the integrated schema there must exist a corresponding (semantically equivalent) element in one of the local schemas. All methods listed above meet this criterion.

◇ *Minimality.* If the same concept is represented in more than one component schema, it must be represented only once in the integrated schema. Redundancy on the schema level must be avoided. Both Model Independent Assertions and Upward Inheritance methods could lead to the state, where the same concept appears in the integrated schema several times. Whereas methods of Formalized Object–Oriented Integration and GIM permit to avoid redundancy, thus meeting the minimality criterion.

◇ *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. All methods listed above meet this criterion. Although, in GIM the occurrences could arise where the relationships between classes in the integrated schema and classes in the schemas to be integrated do not become clear right away.

The choice of the specific integration method is determined by the requirements to the integration quality, given data models, schemas and additional information being at one's disposal. Presently the topic is being researched on the matter of revealing the hybrid and complex methods leading to confining the final result. Nevertheless, none of those methods by its nature is able to take unprecise nondeterministic information as its input. That presents the main contradiction between the pre–integration and the integration methods, because the methods used immediately before integration always deliver the result with uncertainty, which cannot be considered and processed by the integration techniques.

3.4 Data Integration

In contrast to the previous section, which deals with the problem of constructing an integrated schema, this section is devoted to the issues arising when considering source integration at the extensional/instance level.

Real–world data tends to be incomplete, noisy, and inconsistent (for a more detailed conflict classification refer to Section 2.2). Thus, as a first step, special data cleaning tools have to be applied to each source to be integrated in order to fill in the missing values, smooth out noise while identifying outliers, correct inconsistencies in the data, eliminate redundant information. Then, secondly, in order to carry out data integration, it is necessary to identify overlapping data, in particular matching records referring to the same real–world object. This problem is also referred to as the object/instance identity problem that arises when the same real–world object is modelled by different data records. Frequently, the information is only partially redundant and the sources may complement each other by providing additional information about an object. Thus, duplicate information should be purged out and complementing information should be consolidated and merged in order to achieve a consistent view of real–world objects.

Many research papers have been written to the topic of data cleaning, for example, [FLMC01, GFS$^+$01, LLL01, LLLK99, RH01]. Surveys on data cleaning are given in [DJ03, HK01, RD00]. [Kim96] views data cleaning as a process consisting of six steps: elementizing, standardizing, verifying, matching, householding, and documenting. *Elementizing* is another name for data parsing or the procedure of placing elements of a record into the correct fields. *Standardization* brings data elements to forms that are standard throughout the integrated database. *Verification* checks consistency of the standardized data. *Matching* determines whether two records represent data on the same subject. *Householding* combines related records. These stages sometimes make use of auxiliary information. *Documenting* documents the results of the data cleaning steps.

For our purposes the data cleaning process could be combined into a two step process that includes *analysis/detection* and then *correction* of errors and inconsistencies in a data set.

There are two related approaches for data analysis, data profiling and data mining. Data profiling tools provide exhaustive inventory of the data. Data profiling focuses on the instance analysis of individual attributes. It derives information such as the data type, length, value range, discrete values and their frequency, variance, uniqueness, occurrence of null values, typical string pattern, etc., providing an exact view of various quality aspects of the attribute.

Data mining discussed in Section 3.1 helps to discover specific data patterns in data sets. Depending on the type of the data cleaning problem, appropriate data mining methods can be employed. For example, outliers may be detected by clustering, where similar values are organized into groups or clusters. Thus, values that fall outside of the set of clusters may be considered outliers. Data can be smoothed by fitting to a function, such as with regression. Linear regression involves finding the best line to fit two variables, so that one variable could be used to predict the other. Multiple linear regression is an extension of linear regression, where more than two variables are involved and the data is fit to a multidimensional surface. Using regression to find a mathematical equation that fits the data helps smoothing out the noise.

Obviously, data cleaning methods used for cleaning the data and subsequent data integration do not provide exact results, but rather results with a certain level of confidence. Neither do they assume that other methods are being applied to schemas or data prior to them and therefore treat the input information as exact. However, since they do not just analyze the data, but also manipulate it, data cleaning methods utilize various metrics for data quality evaluation. Data quality is a stand–alone problem, which we consider in the next section.

An example of commercial solution on data profiling is *Migration Architect* from Evoke Software Corporation. Migration Architect is a specialized data analysis tool that mines actual data to discover interdependencies between data elements and other inherent data–driven business rules. The uncovered information enables analysts to determine the following metadata for each attribute: data type, length, cardinality, discrete values and their percentage, minimum and maximum values, missing values, and uniqueness. It is extremely useful to accurately map the source files to the consolidated target data structure. Migration Architect consolidates knowledge from multiple systems and rationalizes them into a single, unified, non–redundant model. It keeps track of all the findings and maintains a map of each source–to–target attribute relationship. Migration Architect also generates optimized DDL (Data Definition Language) for any specific RDBMS that may be targeted. The DDL is complete with referential integrity instructions for the model generated.

WizRule (WizSoft Inc.) is an innovative data auditing application that automatically reveals all the rules in a given data and points at the deviations from the set of the discovered rules as suspected errors. WizRule uses a proprietary data mining algorithm based on association rules showing attribute–value conditions that occur frequently together in a given set of data. WizRule can find spelling or unconditional rules, all the “if–then” rules with no limit to their number of clauses and formulae rules. WizRule reads the database once and then automatically separates cases deviating from the rules into data entry errors and suspicious entries. WizRule calculates the degree of deviation from the norm for each field in each record in the database with respect to the discovered rules. Separate reports are generated for data entry (spelling) errors, suspicious entries (deviations), as well as the report of all the rules that govern the database.

Nowadays there are also many commercial data cleaning solutions available on the market that deal with a particular domain, mostly name and address data, for example, Trillium (Trillium Software), PureName PureAddress (Carleton), NADIS (Group1 Software and MasterSoft International) or eliminate duplicates, for example, Centrus Merge/Purge library (Qualitative Marketing Software), matchIT (helpIT Systems Limited), Integrity (Vality), PureIntegrate (Carleton), etc.

3.5 Data Quality

Data quality is a complex and essentially unstructured concept. A major challenge in devising general solutions is that solving data quality problems requires highly domain–specific and context–dependent information, involving interaction with domain experts. Only experts can specify the rules and data flows (dynamic constraints) that are correct. Developing such a set of rules is a critical step in data checking and validation. In addition, the set of specifications that define appropriate data behavior are skewed, in the sense that, while some number of rules can specify say a half of the data, every additional rule specifies smaller and smaller proportions of the data. To cover all the data, we might need hundreds of rules. The ultimate goal of data quality methods as well as metrics is the improved usability and reliability of the data.

Data cleaning, considered in the previous section, improves data quality since it deals with detecting and removing errors and inconsistencies from data. Therefore, data cleaning methods relate to the data quality methods.

In this section we focus on the data quality metrics. The purpose of the data quality metrics is to indicate whether the data is usable and reliable.

We can distinguish five basic data quality metrics:

◇ *Accuracy.* Data must be technically correct, reliable, and free of error. Accuracy is a qualitative assessment of freedom from error. It describes how precise and accurate real–world information is mapped onto local data structures (e.g., exact versus approximate values).

◇ *Consistency.* Data should be presented in the same format and be compatible with previous data. Data is maintained so that it is free of variation or contradiction. Consistency requires that the data stored in a database satisfies the integrity constraints specified for the database.

◇ *Uniqueness* refers to the property that there is one record for each unique entity. Uniqueness shows the ability to establish the uniqueness of a data record (and data key values).

◇ *Timeliness.* The age of the data must be appropriate for the task at hand. Timeliness refers to the fact that the recorded information is up–to–date and not expired.

◇ *Completeness.* Data must be of sufficient breadth, depth, and scope for the task at hand. Completeness means that all real–world information relevant for applications is recorded in the database.

In addition to the basic metrics for improving data quality any other metrics could be employed, for example, the following:

◇ *Accessibility* is the extent to which data is available or easily and quickly retrievable. It measures, for instance, the time between request for access and the actual ability to view the data.

◇ *Interpretability* is the extent to which data is in appropriate languages, symbols, and units, and the definitions are clear. The interpretability metric could be based on availability of metadata (e.g., counting the proportion of fields, tables, keys, and so on, which are documented), and the adherence of data to specifications (e.g., by counting the number of reported problems that are resolved by updating the metadata).

◇ *Understandability* is the extent to which data is easily comprehended.

◇ *Believability* is the extent to which data is regarded as true and credible.

- ◇ *Concise representation* is the extent to which data is compactly represented.
- ◇ *Conformance to schema* is a metric which measures how well a snapshot of the data conforms to the metadata in its schema. For example, are the keys unique, do the values in the fields fit their formats, and so on.

When performing objective assessments, a set of principles should be followed in order to develop metrics specific to one's needs. As a matter of fact a user can employ all of the metrics or only their part, depending on the specific application.

In order to calculate the metrics a set of methods could be made use of, such as, simple ratio, min or max operation, weighted average.

Simple ratio measures the ratio of desired outcomes to total outcomes. Metrics that can use this form are for example, completeness, consistency, concise representation.

Min or max operation could be applied in order to handle metrics that require the aggregation of multiple data quality indicators (variables). One computes the minimum (or maximum) value from among the normalized values of the individual data quality indicators. The min is a conservative whereas max is a liberal form of assessment. An example of metrics that can make use of the min operator is believability. The max operator proves useful in assessing timeliness and accessibility.

Weighted average refers to the multivariate case. When one can specify a degree of importance for each variable, the weighted average could be an appropriate method to use. For example, it could appear a good believability measure.

These three operations are only the most crude approaches in helping to determine the above data quality metrics. When it comes to making a more refined evaluation, the choice of additional methods should be made based on the individual application requirements. The standards for the most common case are being under development. In order to familiarize with the current state of the data quality metrics theory one could refer to [BP85, DM92, Goo95, LSKW02, WS96, Zmu78].

This way, with help of various quality metrics, we can judge how accurate, complete, consistent, etc. the data is. This limits us to acquiring only estimations of usefulness and reliability of the data. The developed quality metrics do not show, however, how well the data in our possession is determined in the sense that during data quality estimation we cannot take into account, that the data had been processed with many other methods and we cannot have an estimate on how that had influenced the data's authenticity.

3.6 Summary

In this chapter a review of existing methods, which can be applied to each step of the integration process, has been given.

If the integration sources are of non– or semi–structural type, then various data mining methods could be used to extract structure (Section 3.1). This area is well explored and currently there is a vast amount of the research approaches as well as commercial solutions, which could be used depending on the data at hand. Data mining methods add the structure information and deliver results with a certain level of confidence.

Schema matching methods, examined in Section 3.2 are concerned with looking for matching elements in given schemas. There is a vast amount of approaches in this area as well, and depending on the input schemas and on the additional information at user's disposal, optimal methods could be chosen, such that, as a result they would output mappings with high validity, but still not exact.

In Section 3.3 the possible schema integration methods are analyzed. This has been a topic of a growing interest in the recent past. There is a number of research approaches, whose classification and ground principles have been given in this section. At that it is important to note that all of them could be quality checked only by such criteria as minimality, correctness, etc. They presuppose deterministic input data and are not able to process uncertain data, which comes into contradiction with methods used to make integration possible, namely, data mining and schema matching.

Questions dealing with the instance level integration are reviewed in Section 3.4. Applicable for that methods are related to data cleaning methods and are well developed to the present moment. However, as well as the methods of schema integration, data cleaning methods do not give a result of deterministic type and also cannot consider non–precise data as an input.

Problems of data quality estimation during the process of data cleaning are described in Section 3.5. The topic of data quality is very actual now and also well developed. And although to this moment the standards defining the quality of the data are not developed, there exist many solutions for improving the data quality and calculating the data quality metrics. All these approaches, however, cannot operate with non–precise data. Their bottom line is that they are being fed with precise input data and the calculation being carried out is to determine how accurate, consistent, unique, etc. the data is.

To summarize the above: all the methods being used by the process of integration give the result with uncertainty and at the same time do not presuppose that the input data is of the uncertain type. Therefore, we see our task not in the development of some integration process improvement methods at some of its steps, but rather in developing the procedure, which would allow to evaluate the whole process, judge about its quality and validity, keeping in mind that each of the used methods introduces additional uncertainty in the whole process of integration. The next chapter classifies uncertainties accumulated in this way during the integration.

4 The Problem of Uncertainty in the Integration Process

In the previous chapters we reviewed a possible integration scenario and also described the methods, which can be used for realizing the integration process. We also have shown that the uncertainty grows in the process of integration. Before we consider building the model that can evaluate uncertainty we find it necessary to give a detailed uncertainty analysis.

For that in the subsequent sections we review the example of data sources integration with absence of uncertainties, i.e., when all information needed for integration is available. After that we present the uncertainty classification. Then we introduce uncertainties into the example and demonstrate how they complicate the process of estimating the integration results. Then we summarize the chapter.

4.1 An Abstract Integration Example

Let us consider a simple integration example in order to study the problem of uncertainty in the integration process. Suppose we should integrate two schemas: schema 1 and schema 2. Suppose each of them contains one superclass with three subclasses as shown in Figure 4.1.

In Section 3.3 we discussed the basic schema integration methods and we also defined the main prerequisites for carrying out the integration process. It is necessary that each of the schemas to be integrated is, firstly, precisely defined and free of contradictions and, secondly, correspondences between schemas must be known.

In the context of our example this means that class A_1 contains three subclasses B_1, C_1, and D_1 with the probability of 100%. The same is true for schema 2, where classes B_2, C_2, and D_2 belong to the class A_2. This way we satisfy the first of the conditions for schema integration, namely, we possess precise information about relationships in each of the integrated sources at the schema level.

To conduct a correct integration the second condition should also be satisfied, i.e., correspondences between schema 1 and schema 2 must be reflected in the integrated schema.

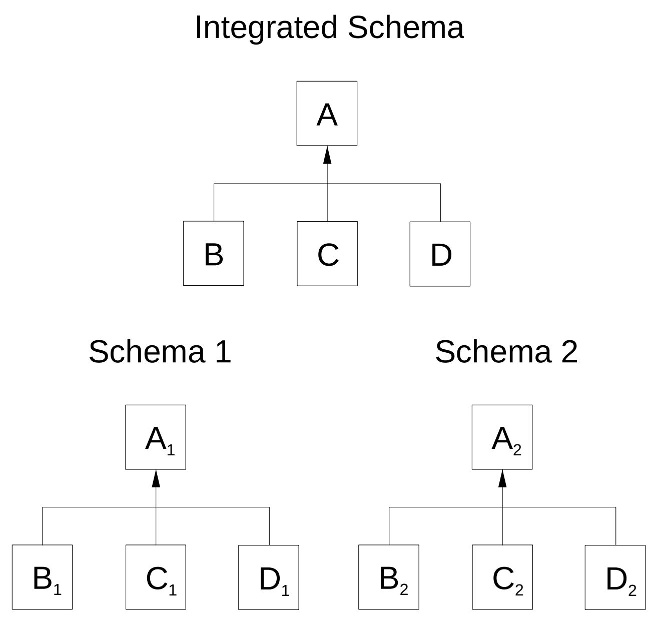

Figure 4.1: Example of the integration process for two schemas

Class correspondences could be of four types: classes could be equivalent, could overlap, one class could be a subclass of another class, and finally classes could be disjoint. Below we review how classes should be integrated depending on the type of correspondences they have.

Suppose that all the classes (A_1 and A_2, B_1 and B_2, C_1 and C_2, D_1 and D_2) are pairwise equivalent. Two classes are equivalent if they both model one real–world class and the objects of one class are equivalent to objects of the other class. It is obvious that in this case any of the two input schemas could be taken as an integrated schema. In our example the integrated schema contains class A with three subclasses B, C, and D, where the following holds $A \equiv A_1 \equiv A_2$, $B \equiv B_1 \equiv B_2$, $C \equiv C_1 \equiv C_2$, and $D \equiv D_1 \equiv D_2$.

Let all the classes (A_1 and A_2, B_1 and B_2, C_1 and C_2, D_1 and D_2) overlap in pairs. Two classes could be considered overlapping if they both model the same real–world class and objects of these two classes intersect, i.e., there exist objects common to both classes and objects that belong only to one of the two classes. Since each of the class pairs that we integrate reflect one real–world class, the integrated schema should look exactly like in the case considered above, namely the integrated schema contains class A with three subclasses B, C, and D. Since the integrated schema has to include all objects (without duplicates) that are contained in the input schemas and correspond to real–world objects, then if classes overlap ($A_1 \cap A_2 \neq \emptyset$, $B_1 \cap B_2 \neq \emptyset$, $C_1 \cap C_2 \neq \emptyset$, and $D_1 \cap D_2 \neq \emptyset$), then integrated classes should be obtained in the following way: $A = A_1 \cup A_2$, $B = B_1 \cup B_2$,

$C = C_1 \cup C_2$, and $D = D_1 \cup D_2$.

The case when one class is a subclass of another class, i.e., one class (superclass) contains all objects of another class (subclass) and additionally contains the objects not contained in the subclass we have already implicitly considered. Classes A_1 and A_2 are the superclasses for classes B_1, C_1, D_1 and B_2, C_2, D_2 respectively. The class–subclass relationships of the source schemas must also be preserved by the integrated schema. Hence, the integrated schema contains class A with the three subclasses B, C, and D.

Two classes could be disjoint, i.e., lacking common objects, but at the same time model the same real–world class. In the terms of integration this corresponds to the case of class overlapping, namely an integrated class must contain the union of the source classes. In our example, if classes are disjoint ($A_1 \cap A_2 = \emptyset$, $B_1 \cap B_2 = \emptyset$, $C_1 \cap C_2 = \emptyset$, and $D_1 \cap D_2 = \emptyset$) then integrated classes should be obtained in the following way: $A = A_1 \cup A_2$, $B = B_1 \cup B_2$, $C = C_1 \cup C_2$, and $D = D_1 \cup D_2$.

Clearly, the considered example does not represent a realistic scenario, but rather an ideal case of integration, because it has been built on an assumption that the integration data is "perfect" and complete information on the semantics, schemas and correspondences both between objects and classes is available.

In the following, we introduce uncertainties and observe how they may change our example.

Suppose precise data about the correspondences between classes B_1 and B_2 is not available. The only knowledge we have is that with probability p these classes model the same real–world class, and with probability q — different real–world classes.

Clearly, with the presence of these probabilities the result of integration could become ambiguous: if during integration it was decided that classes model the same real–world class, then only one class B should be created in the integrated schema (with probability p) for both source classes B_1 and B_2. If we believe that classes represent different real–world classes, then two independent classes B_1 and B_2 (with probability q) must be created in the integrated schemas. This way we have two possible integrated schema, one, in which class B serves as a subclass of class A, and another one, where class A has two independent subclasses B_1 and B_2. At that, in practice, integration methods do not allow to get two integrated schemas, each with its own trustworthiness probability. In respect to that a user has two possible solutions. First one is to use given probabilities only as reference for making local decisions during integration, for example, in our case it could be said that probability q is lower than the chosen threshold and hence we carry out the integration only for the case when classes B_1 and B_2 model the same real–world class. This is the easiest of the approaches and is often used in practice. The problem in this approach is the incapability in estimating the final result, which is undesirable since at the presence of lots of uncertainties and the need to integrate large schemas, the control over the result's trustworthiness diminishes.

The second approach is to compute all possible integration cases and calculate all probabilities. The considered example with only p and q probabilities is trivial, hence let us introduce another uncertainty into it. Let class B_1 in schema 1 belong to class A_1 with probability $p_{B1/A1}$ and class B_2 in schema 2 belong to class A_2 with probability $p_{B2/A2}$. Apparently that if during the integration we believed that classes B_1 and B_2 are carried over to the integrated schema as two independent classes, then the probabilities that these classes belong to the integrated class A stay unchanged, so that class B_1 belongs to class A with probability $p_{B1/A1}$, class B_2 belongs to class A with probability $p_{B2/A2}$. But what should we do in case classes B_1 and B_2 are integrated into a common class B? With what probability does class B belong to class A in the integrated schema?

It becomes obvious that we cannot unambiguously estimate received result without a special mechanism as it was even in the simplest integration case, where intentionally not all types of uncertainties, but rather some of them were introduced. This way we demonstrated that the uncertainties present a problem for integration, because the known integration methods suppose determined input data and do not offer capability to take into account uncertainties during the integration and their influence on the result.

In the next section we discuss existing types of uncertainty and consider their origins.

4.2 Classification of Uncertainty Types

In [AC01] we distinguish three types of uncertainty:

- ◇ Uncertainty about the exact structure of data objects
- ◇ Uncertainty concerning the assignment of data objects to classes
- ◇ Uncertainty concerning the extensional correspondence between classes from two data sources

Uncertainty about the exact structure of data objects. The uncertainty about the exact structure of the data objects means that the schema is not completely known. As mentioned in the previous chapter, one can collect the missing for integration schema information with the help of experts or, for example, by means of employing the data mining techniques. Although data mining techniques are able to partially extract needed information, they do not deliver "perfect" results. This way we can talk about the structural uncertainty that originates from the procedure of assigning structure to semi–structured or non–structured data. It appears at the first integration step in the following way: detecting dependencies and identifying semantics–carrying structure within each source (see Section 2.1). This uncertainty could be measured and could be expressed, for example, through the conditional probability (for more details refer to Section 3.1 of the previous chapter).

Thus, for each data source a set of different schemas could be extracted. In the example shown in Figure 4.2 the object class A includes subclasses B, C, and D with probability p_1 (schema 1). Another possible schema 2 (with probability p_2) defines D as subclass C, etc.

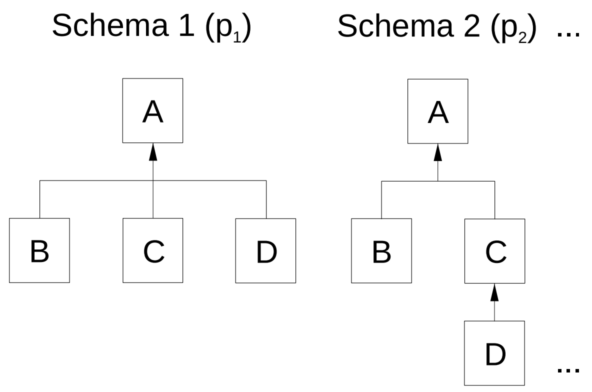

Figure 4.2: Uncertainty about the exact structure of data objects: option 1

Structural uncertainty could be presented in another way, as it is shown in Figure 4.3, when each class belongs to another class with certain probability. Thus, in schema 1 classes B, C, and D belong to the class A with probabilities $p_{1,B}$, $p_{1,C}$, and $p_{1,D}$ respectively. In schema 2 classes B and C belong to class A with probabilities $p_{2,B}$ and $p_{2,C}$. Probability $p_{2,D}$ shows with what probability class D belongs to class C.

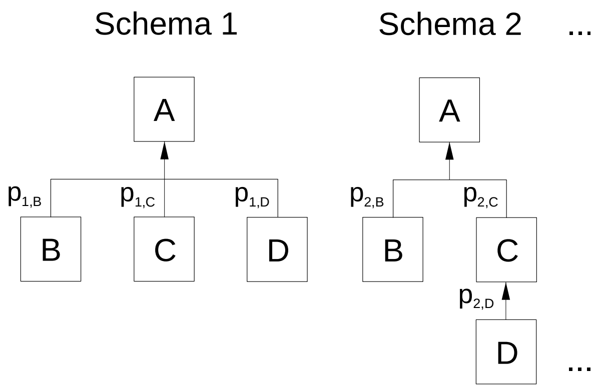

Figure 4.3: Uncertainty about the exact structure of data objects: option 2

Obviously, for integration, for each source only one schema must be chosen that best conforms to the reality. Based on the outputs of the applied structure extraction methods experts can choose the most plausible schema for a given source.

Uncertainty concerning the assignment of data objects to classes. In case when the structure of data objects is not available the uncertainty concerning the assignment of data objects to classes naturally appears. However, even if the structure of the source is well defined this kind of uncertainty could arise due to possible semantic conflicts.

This way of uncertainty could be presented graphically as it is shown in the general case in Figure 4.4, where the data objects ($O_1 \dots O_n$) belong to classes ($A_1 \dots A_m$) with corresponding probabilities ($p_{1,1} \dots p_{1,m}, \dots, p_{n,1} \dots p_{n,m}$).

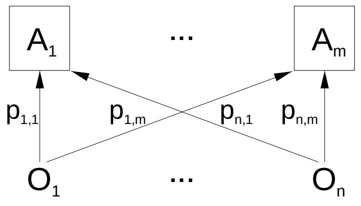

Figure 4.4: Uncertainty concerning the assignment of data objects to classes

Uncertainty concerning the assignment of data objects to classes could be presented in a different way, namely when probabilities correspond to classes and show with what probability the corresponding class was defined correctly. For example, like in using cluster analysis belonging to data mining methods (refer to Section 3.1) when the objects are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity.

The given situation is not directly related to the problem of schema integration, but of instance/data integration. Assigning data objects to classes is an important prerequisite for physical integration, where all objects should be described in terms of one integrated schema. For logical integration the problem appears in the stage of query processing, because every query should be correctly addressed and the result should be correctly interpreted.

Uncertainty concerning the extensional correspondence between classes from two data sources. Another kind of uncertainty occurs if one compares classes of two different data sources in order to get the class/object correspondence. This way it appears at the second integration step: finding the correspondences between the sources on the schema and instance (data) level (see Section 2.1). All types of conflicts described in Section 2.2 may contribute to this uncertainty, which might be investigated, for example, with schema matching and data cleaning methods (see Sections 3.2 and 3.4 respectively).

Let A_1 and A_2 be corresponding classes from two different data sources, $(O_{1,1} \dots O_{1,k})$ and $(O_{2,1} \dots O_{2,k})$ be corresponding objects of these classes. Figure 4.5 presents an example of extensional correspondences between classes from two data sources. A conclusion about correspondences between classes A_1 and A_2 with probabilities of equivalence, inclusion, intersection or disjointness of classes ($p_{\equiv}$, $p_{\supseteq}$, $p_{\cap}$ or $p_{\neq}$ respectively) could be derived from the correspondence probabilities between objects of these classes $(p_1 \dots p_k)$, etc. In some cases probability of class correspondence could be obtained only based on schema information, like in the example when schema–level matchers are used (see Section 3.2).

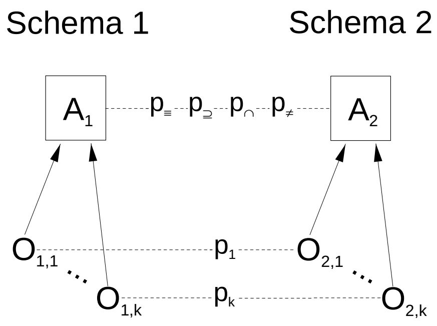

Figure 4.5: Uncertainty concerning the extensional correspondence between classes from two data sources

Thus, we have reviewed existing types of uncertainties. In the next section we analyze an integration example where the numerical values of uncertainties corresponding to the given classification are entered.

4.3 Integration Example with Uncertainties

In practice one rarely meets only one of the uncertainties. Usually one has to deal with the mix of all the uncertainties described above. In Section 4.1 we presented an integration example where uncertainties are completely absent and we also have begun to describe problems, which could appear during the integration due to uncertainties. In this section we give an integration example supplied with the numerical values of uncertainties in conformance with given in the previous section classification. An example of such an integration is given in Figure 4.6.

Suppose we integrate schema 1 and schema 2. Let classes B_1, C_1, D_1 in schema 1 belong to class A_1 with respective probabilities 0.8, 0.8, and 0.7. Besides that, the equivalence probabilities between classes ($p_{\equiv}$) are also known, namely, probability that classes B_1 and B_2 are equivalent is 0.8, probability of equivalence of classes C_1 and

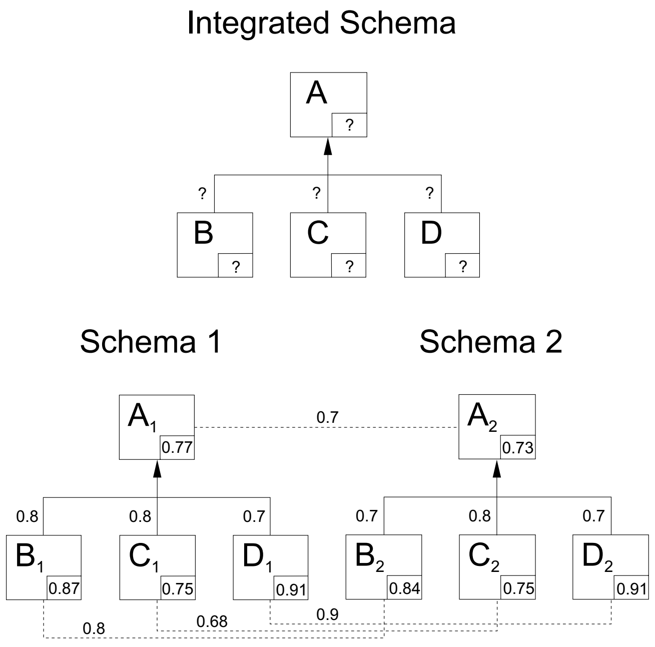

Figure 4.6: Example of uncertainties influence on the estimation of the integration result

C_2 is equal to 0.68, and the equivalence probability of classes D_1 and D_2 is equal to 0.9. Furthermore, the probability that each of the classes was correctly determined is also known, for instance, for the class A_1 that probability comes up to 0.77, for the class A_2 — 0.73, for the class B_2 — 0.84, etc.

As the next step we answer a question about what classes enter the integrated schema. If we define a threshold at 65% for all the probabilities, then apparently integrated schema will have class $A = A_1 \cup A_2$ containing three subclasses $B = B_1 \cup B_2$, $C = C_1 \cup C_2$, and $D = D_1 \cup D_2$. At that it should be noted that as opposed to the example considered in Section 4.1, integrated classes are resulted through the unification of the source classes, in spite of the fact that we consider the case of the source classes equivalence. In this case one should include the class union because in accordance with our data, considered classes are equivalent with certain probability, which supposes presence of the probability that classes to be integrated are disjoint.

Carrying out the integration in this way, the key question is how far we can trust the obtained result. In our example, class C_1 was defined correctly with the same probability as class C_2, thus one can assume, that class C in the integrated schema will also be determined with probability of 0.75. Besides that C_1 and C_2 belong to classes A_1 and A_2 with probability of 80%, hence we can assume that class C belongs to class A also with probability of 0.8. Similar reasoning goes to class D. We assume it is defined at 91% correctly and is a subclass of class A with probability of 70%. In such calculation we do not account for many factors that can influence the resulting probabilities, for example, we do not take account of how the class correspondence probability influences the other probabilities, we do not account for decisions, made by experts during integration. Hence, such approximate calculation of the estimate of the integration result could be sufficient only in the isolated cases when, for example, probabilities of the sources to be integrated coincide and are sufficiently high.

Thus, such calculation could be carried out in the isolated cases and does not provide solution for the general case. For example, how to evaluate the trustworthiness of two classes A and B in the integrated schema? Class B_1 was defined correctly with probability of 0.87 whereas class B_2 with probability of 0.84. Keeping in mind high class correlation, do we have to assume that class B is defined with probability of 0.84 or do we have to take into consideration the size of classes B_1 and B_2 and accept the weighted probability? Another open question is with what probability class A was correctly defined in the integrated schema.

Thus, even on this small example it appears obvious that the evaluation of the integration process is not a trivial task and even at presence of trustworthiness estimates for each employed during the integration method (see Chapter 3) it does not always appear possible to evaluate the trustworthiness of the final result without using special approach. Hence, we find it necessary to develop a mechanism that would allow to evaluate the whole integration process independently of uncertainty types present in it and the nature of these uncertainties. In our work, for evaluating the integration process, we propose to use statistical methods. Next chapter is devoted to reviewing and choosing the certain statistical method that would fit the integration process.

4.4 Summary

In the given chapter, we have demonstrated, using a small example, how the data can be integrated under a condition that it is completely defined and we possess the whole information about the semantics, schemas, and correspondences between schemas.

Such situation cannot be real, because in practice various uncertainties are usually present and not the whole information needed for integration is available.

Thus, we find it important to classify the uncertainties that can appear in the process of integration. We distinguish three types of uncertainty: uncertainty about the exact structure of data objects, uncertainty concerning the assignment of data objects to classes, and uncertainty concerning the extensional correspondence between classes from two data sources.

Having introduced the uncertainties into the integration example we demonstrated the impossibility of calculating the uncertainty using integration methods, which leads to contradiction between methods of integration and the real–world conditions. Thus, it appears that the probabilities could serve as the input data, but the methods, allowing to account for these probabilities and their influence on the final result, are not developed. Thus, uncertainty presents a problem for integration.

We find that it is important to develop a transparent method for evaluating uncertainties in the process of integration.

5 Latent Variable Model

In this chapter we exploit the Latent Variable Model. The general idea and basic principles are given in Section 5.1. The theoretical framework of the concrete LCA method relevant for our use is considered in Sections 5.2 and 5.3. In those sections two approaches for building the latent model are examined: estimation method based on simultaneous linear equations and maximum likelihood estimation method. The goodness of fit test for evaluating the built model is explained in Section 5.4. Section 5.5 summarizes this chapter.

5.1 Principles of Latent Variable Model

Latent variable models provide an important tool for the analysis of multivariate data. They offer a conceptual framework within which many disparate methods can be unified and a base upon which new methods can be developed. Latent variable models include such methods as factor analysis, latent class analysis, latent trait analysis, latent profile analysis, etc. [BK99, Bas94, Goo74, LH68, McC87].

Latent variable models are based on the statistical model and are used to study the patterns of relationship among many dependent variables, with the goal of discovering something about the nature of the latent (unobservable) variables that affect them, even though those latent variables were not measured directly. The latent variables are called factors. Dependent variables used in Latent Variable Models are manifest variables which can be directly observed.

Generally, both latent and manifest variables could be metrical or categorical. Metrical variables have the values in the set of real numbers and may be discrete or continuous. Categorical variables assign individuals to one of a set of categories.

The relevant method for the integration task is Latent Class Analysis (LCA), since both latent and manifest variables are categorical.

A typical LCA model suggests answers to three major questions:

1. How many different factors are needed to explain the pattern of relationships among the variables?

2. How well do the hypothesized factors explain the observed data?

3. How much purely random or unique variance does each observed variable include?

In the next section we present the LCA model formally.

5.2 Theoretical Framework of Latent Class Analysis

In this section we adopt the definitions given in [Bas94]. We extend them by correcting some indices and by introducing some additional notions, necessary for our application.

The latent class model can be understood and developed from standard probability theory making use of Bayes' theorem. Consider any two random variables x_1 and x_2 defined as

$$x_i = \begin{cases} 1, & \text{if event } E_i \text{ is observed} \\ 0, & \text{if event } E_i \text{ is not observed} \end{cases}$$

for $i = 1$ and 2. The outcome corresponding to the code "1", which is defined arbitrarily, is known as the "positive" outcome or the "success" of the trial and the code "0" is termed as the "negative" or the "failure" outcome of the trial. For example, x_i may represent independent binomial trials or outcomes for any other discrete distribution. The following theorem is well known from the classical probability theory.

Theorem 5.1. Let $A_1, A_2, \ldots, A_m$ represent a mutually exclusive partition of a sample space S, such that $P(A_s) \neq 0$, $s = 1, 2, \ldots, m$, and $\sum_{s=1}^{m} P(A_s) = 1$. Let E_i be some arbitrary event defined in S. Then we have

$$P(E_i) = \sum_{s=1}^{m} P(A_s) P(E_i|A_s). \tag{5.1}$$

The result of the Theorem 5.1 can be generalized to any finite number of independent events. Thus for two events we have

$$P(E_i \cap E_j) = \sum_{s=1}^{m} P(E_i \cap E_j \cap A_s),$$

where

$$P(E_i \cap E_j | A_s) = \frac{P(E_i \cap E_j \cap A_s)}{P(A_s)}$$

and cross multiplying yields

$$\begin{aligned} P(E_i \cap E_j) &= \sum_{s=1}^{m} P(A_s) P(E_i \cap E_j | A_s) \\ &= \sum_{s=1}^{m} P(A_s) P(E_i|A_s) P(E_j|A_s). \end{aligned} \tag{5.2}$$

For any finite number of k arbitrary events $E_1, E_2, \ldots, E_k$, we have the relation

$$P(E_1 \cap E_2 \cap \ldots \cap E_k) = \sum_{s=1}^{m} P(A_s)P(E_1|A_s)P(E_2|A_s)\ldots P(E_k|A_s). \tag{5.3}$$

In a typical application of equation (5.3) the probabilities $P(A_s)$ of the partitioning events are given, and $k < m$. In the latent class model however we assume the events $A_1, A_2, \ldots, A_m$ to be unobservable and the probabilities $P(A_s)$ to be unknown. The problem then poses itself as follows. Given the observed probabilities $P(E_1), P(E_2), \ldots, P(E_k)$ is it possible to compute the conditional probabilities $P(E_i|A_s)$ $(i = 1, 2, \ldots, k; s = 1, 2, \ldots, m)$ together with the partition probabilities $P(A_s)$ such that $m < k$? Consider k dichotomous random variables (the observed classes or categories) and $m < k$ unobserved classes, where

$P(A_s) = \pi_s$	–	the unobserved probability of being in the sth latent class, $s = 1,\ 2,\ \ldots\ , m$;
$P(E_i) = p_i$	–	observed proportion of sample points that respond positively to the ith category $(i = 1,\ 2,\ \ldots\ , k)$;
$P(E_i \cap E_j) = p_{ij}$	–	observed proportion of sample points that respond positively to both the ith and jth categories $(i \neq j,\ p_{ij} = p_{ji})$;
$P(E_i \cap E_j \cap \ldots E_k) = p_{ij\ldots k}$	–	observed proportion of sample points that respond positively to the ith, jth, …, kth categories $(i \neq j \neq \ldots \neq k\ )$ where permutations of indices are excluded;
$P(E_i|A_s) = \nu_{is}$	–	the unobserved conditional probability that a sample point in the sth latent class is also in the ith observed category.

Since the sets $A_1, A_2, \ldots, A_m$ represent a partition of the sample space, $\sum_{s=1}^{m} \pi_s = 1$. Using the notation shown above we have the normal equations

$$1 = \sum_{s=1}^{m} \pi_s \tag{5.4}$$

$$p_i = \sum_{s=1}^{m} \pi_s \nu_{is} \quad (i = 1, 2, \cdots, k) \tag{5.5}$$

$$p_{ij} = \sum_{s=1}^{m} \pi_s \nu_{is} \nu_{js} \quad (i, j = 1, 2, \cdots, k) \tag{5.6}$$

$$p_{ij\cdots l} = \sum_{s=1}^{m} \pi_s \nu_{is} \nu_{js} \cdots \nu_{ls} \qquad (i, j, \cdots, l = 1, 2, \cdots, k) \tag{5.7}$$

where $i \neq j \neq \cdots \neq l$ and permuted subscripts do not appear. Equations (5.4) – (5.7) express observed probabilities in terms of unknown probabilities, and represent the general system of normal equations for a latent class model. They are also known in the literature as the "accounting" equations. In practice, given a sample the observed joint frequencies or the manifest probabilities are substituted on the left–hand side of equations (5.4) – (5.7), and assuming the existence of a unique solution (identifiability) the system can be solved to yield estimates of the unobserved parameters π_s, ν_{is}, $\nu_{js}, \ldots, \nu_{ls}$, known as the latent probabilities.

For $k > m$ observed categories, the largest possible number of joint probabilities is when $l = k$. Taking $p_0 \equiv 1$ as the "null subscript" probability, the maximum number of equations which is possible is then $\sum_{l=0}^{k} C_k^{(l)} = 2^k$. Since for k observed categories and $m < k$ latent classes the total number of unknown latent parameters is $m + mk = m(k+1)$, a necessary condition for identifiability is that $2^k \geq m(k+1)$, that is,

$$\frac{2^k}{k+1} \geq m. \tag{5.8}$$

Equation (5.8) can always be satisfied since given m we can always choose k sufficiently large assuming large sample be given. Conversely, if the system is identifiable and if there are more equations (i.e., joint probabilities) than parameters, a solution can always be found by using a subset of the normal equations.

According to equations (5.4) – (5.7) and condition (5.8) the minimal case which the LCA model is able to consider is a case where $k = 3$ and $m = 2$. Then the LCA model could be represented as the following system of eight equations with eight variables.

$$\left.\begin{aligned}
1 &= \pi_1 + \pi_2 \\
p_1 &= \pi_1\nu_{11} + \pi_2\nu_{12} \\
p_2 &= \pi_1\nu_{21} + \pi_2\nu_{22} \\
p_3 &= \pi_1\nu_{31} + \pi_2\nu_{32} \\
p_{12} &= \pi_1\nu_{11}\nu_{21} + \pi_2\nu_{12}\nu_{22} \\
p_{13} &= \pi_1\nu_{11}\nu_{31} + \pi_2\nu_{12}\nu_{32} \\
p_{23} &= \pi_1\nu_{21}\nu_{31} + \pi_2\nu_{22}\nu_{32} \\
p_{123} &= \pi_1\nu_{11}\nu_{21}\nu_{31} + \pi_2\nu_{12}\nu_{22}\nu_{32}
\end{aligned}\right\} \tag{5.9}$$

As mentioned above, for practical calculations, it is sometimes convenient to use observed frequencies of the patterns as opposed to the manifest probabilities. Then a contingency table could be built. The Contingency Table 5.1 reflects the minimal possible case scenario ($k = 3$ and $m = 2$), where every pattern $i - j - k$ $(i, j, k = 0, 1)$ is counted n_{ijk} times and the corresponding observed frequencies are $f_{ijk} = n_{ijk}/n$, where $n = \sum_{ijk} n_{ijk}$ is a total number of sample points.

Table 5.1: Contingency table

Observed Pattern	**Number of occurrences**	**Observed frequency**
$1-1-1$	n_{111}	f_{111}
$1-1-0$	n_{110}	f_{110}
$1-0-1$	n_{101}	f_{101}
$1-0-0$	n_{100}	f_{100}
$0-1-1$	n_{011}	f_{011}
$0-1-0$	n_{010}	f_{010}
$0-0-1$	n_{001}	f_{001}
$0-0-0$	n_{000}	f_{000}

Obviously, a set of $\{f_{ijk}\}$ values can be easily derived from a set of seven values $\{p_i, p_{ij}, p_{ijk}\}$:

$$\left.\begin{array}{rcl} f_{111} &=& p_{123} \\ f_{110} &=& p_{12} - p_{123} \\ f_{101} &=& p_{13} - p_{123} \\ f_{100} &=& p_1 + p_{123} - p_{13} - p_{12} \\ f_{011} &=& p_{23} - p_{123} \\ f_{010} &=& p_2 + p_{123} - p_{23} - p_{12} \\ f_{001} &=& p_3 + p_{123} - p_{23} - p_{13} \\ f_{000} &=& 1 - p_{123} + p_{23} + p_{13} + p_{12} - p_1 - p_2 - p_3 \end{array}\right\} \tag{5.10}$$

Conversely, a set of seven values $\{p_i, p_{ij}, p_{ijk}\}$ can be computed given set of $\{f_{ijk}\}$ values:

$$\left.\begin{array}{rcl} p_1 &=& f_{111} + f_{110} + f_{101} + f_{100} \\ p_2 &=& f_{111} + f_{110} + f_{011} + f_{010} \\ p_3 &=& f_{111} + f_{101} + f_{011} + f_{001} \\ p_{12} &=& f_{111} + f_{110} \\ p_{13} &=& f_{111} + f_{101} \\ p_{23} &=& f_{111} + f_{011} \\ p_{123} &=& f_{111} \end{array}\right\} \tag{5.11}$$

When the observed categories are binomial the method of maximum likelihood can be used to estimate the latent probabilities.

5.3 Maximum Likelihood Estimation

In this subsection we consider the method of maximum likelihood estimation described in [BK99] in the context of the definitions of the previous subsection.

Suppose there are k binary variables $x_1, x_2, \dots, x_k$ with $x_i = 0$ or 1 for all i. This collection can be presented as a vector $\mathbf{x} = (x_1, x_2, \dots, x_k)$. Let us consider whether their mutual association could be accounted for by a single binary variable s. In other words, is it possible to divide the population into two parts so that the x are mutually independent in each group? The prior distribution of s may then be written

$$h(1) = P\{s = 1\} = \pi \text{ and } h(0) = 1 - h(1). \tag{5.12}$$

The conditional distribution of x_i given s, $g_i(x_i|s)$ will be that of a Bernoulli random variable written

$$g_i(x_i|s) = P\{x_i|s\} = \nu_{is}^{x_i}(1-\nu_{is})^{1-x_i} = \begin{cases} \nu_{is}, & \text{if } x_i = 1 \\ 1-\nu_{is}, & \text{if } x_i = 0 \end{cases} \quad (x_i, s = 0, 1), \tag{5.13}$$

where ν_{is} is the probability that $x_i = 1$ when the latent class is s. Notice that in this simple case the form of the distribution h and $\{g_i\}$ is not in question; it is only their parameters, $\{\pi\}$, $\{\nu_{i1}\}$ and $\{\nu_{i0}\}$ which are unspecified by the model.

As only $\mathbf{x}$ can be observed, any inference must be based on joint distribution whose density is denoted $f(\mathbf{x})$. Thus, the LCA model with binary manifest variables and a single binary latent variable could be represented as follows:

$$f(\mathbf{x}) = \pi \prod_{i=1}^{k} \nu_{i1}^{x_i}(1-\nu_{i1})^{1-x_i} + (1-\pi)\prod_{i=1}^{k} \nu_{i0}^{x_i}(1-\nu_{i0})^{1-x_i}. \tag{5.14}$$

The extension of the two–class to a m–class model is almost immediate. Let ν_{is} be the probability of a positive response on variable i for an object in latent class s $(i = 1, 2, \dots, k;\ s = 0, 1, \dots, m-1)$ and let π_s be the prior probability that a randomly chosen object is in class s $(\sum_{s=0}^{m-1} \pi_s = 1)$. For the case of m classes (5.14) becomes

$$f(\mathbf{x}) = \sum_{s=0}^{m-1} \pi_s \prod_{i=1}^{k} \nu_{is}^{x_i}(1-\nu_{is})^{1-x_i}. \tag{5.15}$$

The posterior probability that an object with response vector $\mathbf{x}$ belongs to latent class s is thus

$$h(s|\mathbf{x}) = \pi_s \prod_{i=1}^{k} \nu_{is}^{x_i}(1-\nu_{is})^{1-x_i} / f(\mathbf{x}) \quad (s = 0, 1, \dots, m-1). \tag{5.16}$$

From (5.15) we find the log–likelihood function for a random sample of size n to be

$$L = \sum_{h=1}^{n} \ln \left\{ \sum_{s=0}^{m-1} \pi_s \prod_{i=1}^{k} \nu_{is}^{x_{ih}} (1-\nu_{is})^{1-x_{ih}} \right\}. \tag{5.17}$$

This has to be maximized subject to $\sum \pi_s = 1$, so we find the unrestrained maximum of

$$\phi = L + \theta \sum_{s=0}^{m-1} \pi_s,$$

where θ is an undetermined multiplier. Finding partial derivatives, we have

$$\begin{aligned} \frac{\partial \phi}{\partial \pi_s} &= \sum_{h=1}^{n} \left\{ \prod_{i=1}^{k} \nu_{is}^{x_{ih}} (1-\nu_{is})^{1-x_{ih}} / f(\mathbf{x}_h) \right\} + \theta \qquad (s = 0, 1, \ldots, m-1) \\ &= \sum_{h=1}^{n} \{ g(\mathbf{x}_h | s) / f(\mathbf{x}_h) \} + \theta, \end{aligned} \tag{5.18}$$

where $g(\mathbf{x}_h|s)$ is the joint probability of $\mathbf{x}_h$ for an object in class s. Also,

$$\frac{\partial g(\mathbf{x}_h|s)}{\partial \nu_{is}} = \sum_{h=1}^{n} \pi_s \frac{\partial}{\partial \nu_{is}} g(\mathbf{x}_h|s)/f(\mathbf{x}_h) \qquad (i = 1, 2, \ldots, k;\ s = 0, 1, \ldots, m-1).$$

Now

$$\begin{aligned} \frac{\partial g(\mathbf{x}_h|s)}{\partial \nu_{is}} &= \frac{\partial}{\partial \nu_{is}} \exp \left(\sum_{i=1}^{k} \{ x_{ih} \ln \nu_{is} + (1-x_{ih}) \ln(1-\nu_{is}) \} \right) \\ &= g(\mathbf{x}_h|s) \left\{ \frac{x_{ih}}{\nu_{is}} - \frac{1-x_{ih}}{1-\nu_{is}} \right\} \\ &= (x_{ih} - \nu_{is}) g(\mathbf{x}_h|s) / \nu_{is}(1-\nu_{is}). \end{aligned} \tag{5.19}$$

Therefore,

$$\frac{\partial \phi}{\partial \nu_{is}} = \{ \pi_s / \nu_{is}(1-\nu_{is}) \} \sum_{h=1}^{n} (x_{ih} - \nu_{is}) g(\mathbf{x}_h|s) / f(\mathbf{x}_h). \tag{5.20}$$

The resulting equations can be simplified by expressing (5.18) and (5.20) in terms of the posterior probabilities $\{h(s|\mathbf{x}_h)\}$. By Bayes' theorem,

$$h(s|\mathbf{x}_h) = \pi_s g(\mathbf{x}_h|s) / f(\mathbf{x}_h). \tag{5.21}$$

Substituting in (5.18) and setting equal to zero, we find

$$\sum_{h=1}^{n} h(s|\mathbf{x}_h) = -\theta\pi_s.$$

Summing both sides over s and using $\sum \pi_s = 1$ gives $\theta = -n$, and hence the first estimating equation is

$$\hat{\pi}_s = \sum_{h=1}^{n} h(s|\mathbf{x}_h)/n \qquad (s = 0, 1, \dots, m-1). \tag{5.22}$$

The second is

$$\sum_{h=1}^{n} (x_{ih} - \nu_{is}) h(s|\mathbf{x}_h)/\nu_{is}(1-\nu_{is}),$$

hence

$$\begin{aligned} \hat{\nu}_{is} &= \sum_{h=1}^{n} x_{ih} h(s|\mathbf{x}_h) \Big/ \sum_{h=1}^{n} h(s|\mathbf{x}_h) \\ &= \sum_{h=1}^{n} x_{ih} h(s|\mathbf{x}_h)/n\hat{\pi}_s \qquad (i = 1, 2, \dots, k;\ s = 0, 1, \dots, m-1). \end{aligned} \tag{5.23}$$

Although these equations have a simple form it must be remembered that $h(s|\mathbf{x}_h)$ is a complicated function of $\{\pi_s\}$ and $\{\nu_{is}\}$ given by

$$h(s|\mathbf{x}_h) = \pi_s \prod_{i=1}^{k} \nu_{is}^{x_{ih}} (1-\nu_{is})^{1-x_{ih}} \Big/ \sum_{s=0}^{m-1} \pi_s \prod_{i=1}^{k} \nu_{is}^{x_{ih}} (1-\nu_{is})^{1-x_{ih}}. \tag{5.24}$$

However, if $h(s|\mathbf{x}_h)$ were known it would be easy to solve (5.22) and (5.23) for $\{\pi_s\}$ and $\{\nu_{is}\}$. The Expectation Maximization (EM) algorithm takes advantage of this fact proceeding in a "zig–zag" fashion as follows:

(i) Choose an initial set of posterior probabilities $\{h(s|\mathbf{x}_h)\}$.

(ii) Use (5.22) and (5.23) to obtain a first approximation to $\{\hat{\pi}_s\}$ and $\{\hat{\nu}_{is}\}$.

(iii) Substitute these estimates into (5.24) to obtain improved estimates of $\{h(s|\mathbf{x}_h)\}$.

(iv) Return to (ii) to obtain second approximations to the parameters and continue the cycle until convergence is attained.

The solution reached will be a local maximum. It is known that models of this kind may have multiple maxima and the risk of this appears to increase as m, the number of classes, increases and to decrease with increasing sample size. By using different starting values one can guard against the risk of mistaking a local for a global maximum, but if such multiple maxima do occur it is not clear what interpretation should be placed on the different set of latent classes implied by the various local maxima.

A reasonable way of starting the iteration is to allocate objects, arbitrarily, to latent classes on the basis of their total score ($\sum_{i=1}^{k} x_i$) that is, to take $\{h(s|\mathbf{x}_h) = 1\}$ if $\mathbf{x}_h$ is allocated to class s and $\{h(s|\mathbf{x}_h) = 0\}$ otherwise. Although the method may take a very large number of iterations to converge, the steps are simple and fast, so the total computing time is unlikely to be excessive. As well as providing parameter estimates the method also provides the posterior probabilities that each object belongs to a given latent class.

Nowadays there are many software packages available that implement the above given theory, for instance, such programs as PANMARK (Assessment Systems Corporation), Latent GOLD (Statistical Innovations Inc.), LEM (freeware program developed by Jeroen Vermunt, Department of Methodology, Faculty of Social Sciences, Tilburg University), LATCLASS (supporting software for the [BK99]), etc. In our work we use the freeware program WinLTA developed by the Methodology Center of the Pennsylvania State University.

5.4 Goodness of Fit

For evaluating the latent class model the goodness of fit test could be applied. Goodness of fit is assessed by comparing the observed and predicted response pattern frequencies and can be carried out using the two test statistics, namely, the likelihood ratio usually denoted G^2 and the Pearson chi–squared statistics denoted χ^2.

$$\chi^2 = \sum \frac{\left(f_{ij\ldots k} - \hat{f}_{ij\ldots k}\right)^2}{\hat{f}_{ij\ldots k}}. \tag{5.25}$$

$$G^2 = 2 \sum f_{ij\ldots k} \ln(f_{ij\ldots k}/\hat{f}_{ij\ldots k}). \tag{5.26}$$

In the equations (5.25) and (5.26) $i, j, \cdots, k$ could take only ones for the positive and zeros for the negative outcomes of the trial, $f_{ij\ldots k}$ represents the observed frequency of the response pattern $ij\ldots k$, and $\hat{f}_{ij\ldots k}$ represents the frequency predicted by the model.

Asymptotically, G^2 is distributed as a chi–square (χ^2) with degrees of freedom equal to

$$2^k - km - (m-1),$$

where k is number of categories, 2^k is the maximum number of equations possible in the LCA model, km is number of response probabilities, and m is number of latent classes.

The value given by (5.25) or (5.26) should not exceed the critical value χ_0^2 that could be obtained from the table of critical values of the χ^2 distribution. For example, for the model where $k = 3$ and $m = 2$ we could choose a critical region of size 0.05 (permissible error probability is 5%), then the critical value of the χ^2 distribution with one degree of freedom will be $\chi_0^2 = 3.841$. The resulting model is then considered as fit if the value of χ^2 or G^2 is less or equal to 3.841.

If the model was rejected by the goodness of fit test, then the parameters of the latent model should be changed until the values G^2 or χ^2 are acceptable.

5.5 Summary

In this chapter the overview of the Latent Variable Models has been given. Among its methods, the one relevant for the integration task, Latent Class Analysis (LCA) has been presented. In the following the theory of the LCA method is summarized.

The LCA is a statistical method that is used to discover the latent unobserved variables (factors), which best describe relationships between manifest observed variables.

The LCA defines latent classes so that, within each latent class, each variable is statistically independent of every other variable. To say this differently, latent classes are defined such that, if one removes the effect of latent class membership on the data, all that remains is randomness (understood here as complete independence among measures).

The LCA provides a means for testing whether latent factors explain the observed pattern of relationships among the variables, how many different factors are needed to explain the observed data, the substantive meaning of the latent variables, the prominence of the manifest variables as indicators of latent factors, and how much purely random variance does each observed variable include.

The LCA supposes a simple parametric model and uses observed data to estimate parameter values for the model. The model parameters are: probability that a sample point is in the latent class and conditional response probability that a sample point that belongs to the latent class is also in the observed category. The input data for the LCA are manifest probabilities or observed frequencies of the patterns that could be presented in the form of the contingency table.

The LCA uses estimation method based on simultaneous linear equations (refer to Section 5.2) or simple iterative proportional fitting could be used to find maximum likelihood parameter values; this method is type of algorithm described in Section 5.3.

Once the latent class model is estimated, it need to be verified in terms of their goodness of fit. Model fit is assessed by comparing the observed cross classification frequencies to the expected frequencies predicted by the model. The difference is formally assessed with a likelihood ratio or the Pearson chi–squared statistic (refer to Section 5.4). This statistics shows which latent categorization explains the original association or, in other words, tells us which of the initial test latent models is the best fit for our problem.

In the next chapter we demonstrate how the LCA is used in the integration task.

6 Applying Latent Class Analysis to the Integration Task

In this chapter we come to the objective of our work. We demonstrate how the integration process and the statistical methods could be combined together. In general it appears hard to comprehend how these two, operating on values of absolutely different nature could be combined at all. It is clearly seen that the information the integration task is dealing with is discrete on the example of the objects, which either belong to classes or do not, partial membership of objects in the classes is clearly a non–existing terminology. Integration methods do not operate with uncertainties, they suppose complete knowledge about belonging of objects to classes, the class correspondences, etc.

However, since we need to integrate unstructured or semi–structured data we have to employ special methods, which extract structure and determine correspondences and dependencies within and between data sources. Therefore, we no longer deal with the real–world classes and real–world objects, which results into uncertainties appearing in the integration process. We now describe how exactly the integration process should be approached in such a way that it becomes a direct input data for statistics, and how conclusions about the integration process can be drawn from the results of the statistics methods.

Thus, we propose analyzing the uncertainties and their propagation from one integration step to another using the available statistical methods. The principle of the Latent Variable Models is based on revealing knowledge about the latent variables using variables that can be observed. The integration process has a similar pattern, that is there are some absolute or real–world entities, not directly visible and there are entities at the outcome of the integration that we observe directly and based on which we would like to give an estimation to the invisible ones, that they describe. The correlation between the statistical method and the integration process is obvious. Our task is to use this similarity and express the integration process in terms of the statistical model such that we can get a grip on the statistical estimation of the real–world entities of the integration process.

Building a statistical model of the integration task is considered in Section 6.1. In Section 6.2 we demonstrate on examples how the integration result can be evaluated using the LCA. Section 6.3 considers questions related to integration quality. Section 6.4 concludes the chapter.

6.1 Statistical Model of the Integration Task

In the previous chapter we reviewed the theory of the Latent Variable Models and suggested that the LCA could be used for estimating the integration process. In order to achieve that, a statistical model of the integration task should be built. We distinguish the following steps for converting integration task into the statistics plane:

- ◇ Data representation in the integration task
- ◇ Construction of the class membership table
- ◇ Integration task in terms of the statistical model

Data representation in the integration task. Taking into account that the LCA uses a contingency table as an input data, we need to represent the data of the integration task in the same way, so that the LCA is able to work on it. According to Section 2.1 for integrating k data sources, first the correspondences of schema nodes of these data sources should be determined, so that the integrated schema can be built. Then for carrying out the data integration the correspondences between the objects from corresponding schema nodes (class, subclass) have to be established. Thus, for each schema node we have a set of n objects and a knowledge (partly uncertain) about their class membership. This information could be presented as a table of the class membership of objects in each data source. We denote the class membership with the following values:

$$x_{i,j} = \begin{cases} 0, & \text{if object does not belong to the class} \\ 1, & \text{if object belongs to the class} \\ 2, & \text{missing data} \end{cases}$$

Table 6.1 (where subscript $i = 1, 2, \ldots, n$ refers to the object number, $j = 1, 2, \ldots, k$ is the data source) is an illustration of the class membership table in general case. A set of values $x_{i,j}$ forms the n–rows by k–columns $(n \times k)$ matrix X of class membership.

In the case that we possess exact knowledge about membership of objects in classes, x_{ij} take only values “1” and “2”. The exact meaning of these values is as following: “1” means that the considered object belongs to the considered class in the corresponding data source, “2” means that the considered object does not exist in the corresponding class of the corresponding data source. Thus, supposing that all columns of the class membership table contain only “1”s, means that the considered classes from given data

Table 6.1: Class membership table

i \ j	**1**	**2**	...	**k**
1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,k}$
2	$x_{2,1}$	$x_{2,2}$	...	$x_{2,k}$
...	...	...	...	...
n	$x_{n,1}$	$x_{n,2}$	...	$x_{n,k}$

sources are equivalent. The case when all columns of the class membership table contain only “2”s corresponds to a situation when the considered classes from given data sources are disjoint. Table containing both ones and twos represents overlapping classes, i.e., there exist objects common to all classes and objects that belong only to certain classes.

As a consequence of applying information extraction methods we have neither exact knowledge about membership of objects in classes nor correspondences between classes, i.e., both class membership and correspondences are annotated by a certain level of confidence, support, etc. At that, if the support value is different from 1 it cannot be inserted into the table instead of the value x_{ij}. That is why some objects will be introduced by a zero and some by a one, but which is which is not exactly known. Support of class membership will be represented in the table by a ratio of the number of “1”s to n column wise, whereas support of class correspondence will be represented by a ratio of pairs “1–1” to n. It is important that in such an interpretation the row–object relationships will be lost.

Besides, the presence of the support values in the table implies that there could be mistakes of two types, namely when an object belongs to the class but was defined as “0” and when an object not belonging to the class was defined as “1”. These mistakes introduce uncertainty into the class membership table so that the rows with random combinations of “1” and “0” appear, which means that the rows with all ones and all zeros could appear as well. Therefore, this table is a collection of both correct and random data. All three types of uncertainties described in Section 4.2 contribute to the values $x_{i,j}$. Thus, the random part cannot be mechanically separated from the rest of the table. A statistical method should be applied for that.

Statistical methods do not require assigning a certain row to a certain object. That is why the rows can be sorted in order to compose a clear structure of the table. It is convenient to distinguish three different parts in the table, namely, the definite part (I), which includes the objects that are presented in each data source (equivalence), a part (II), which includes missing data (disjoint), and a so–called random part (III), which includes all the uncertainties. The Table 6.2 reflects the situation described above and presents the three parts of the class membership table.

Table 6.2: Three parts of the class membership table

i \ j	**1**	**2**	...	**k**	
1	1	1	...	1	**I**
2	1	1	...	1	
...	...	...	...	...	
...	1	1	...	1	
...	2	1	...	1	**II**
...	1	1	...	2	
...	...	...	...	...	
...	1	2	...	1	
...	1	0	...	1	**III**
...	0	1	...	0	
...	...	...	...	...	
n	1	1	...	0	

Data representation as given in the Table 6.2 is not necessary for the LCA, but it gives a descriptive representation for understanding LCA's principle of work, namely separation of the random part from all the other ones.

Construction of the class membership table. Since the data sources that we need to integrate do not possess defined structure and the exact correspondences between them are also not known, we construct the class membership table based on the data obtained by various methods used in the integration process. In fact it is not possible to identify a certain object with a certain row. The table should be reconstructed using the integral parameters delivered by the applied techniques such as data mining, schema matching, data cleaning, etc.

The support of the object's class membership for each data source is provided by data mining techniques (see Section 3.1). Therefore, the proportion of values "0" and "1" in every column of the class membership table is known.

Schema matching (refer to Section 3.2) delivers the support of class correspondence. We use this information together with the information obtained by data mining and data cleaning to find the parts of the table, which should be disjoint for every pair of corresponding classes in different data sources.

The value of the integral parameter (for example, support of class membership) means that the corresponding column of the table contains a certain number of "1"s and "0"s, but there is no information which row contains "1" and which row contains "0". Therefore, we can sort the values in every column in order to fit the value of the support of the class correspondence. This integral parameter is responsible for the relative number of combinations "1–0", "0–1", "1–1" and "0–0" for every two columns of the table.

This information may not be sufficient for the complete reconstruction of the class membership table, but this is not really needed for applying statistical methods as soon as the integral parameters are given.

Integration task in terms of the statistical model. In Section 5.2 the theory of LCA was reviewed. As a next step we would like to analyze similarities and differences between the data representation in integration and statistics. In addition to that we would like to draw correspondences between the concepts of LCA and integration.

The information we deal with in the integration process has a discrete nature. Every real object is unique and cannot have a partial class membership. As described above, the integration task operates with only such boolean statements as, that an object is either a real–world object or not, it either belongs to real–world class or it does not. On the contrary, the statistics does not make any difference between objects and considers a class membership of each object as a single measurement (or trial) of the random variable assigned to this class.

Keeping this in mind, the term sample in statistics corresponds to a set of objects, part of which represents real–world objects, whereas another part represents uncertainties. Uncertainties appear in a sample as a consequence of applying the knowledge extraction methods representing an error of two types: when the real–world objects were not revealed and when those revealed are not the real–world objects.

Thus, the concept of sample point corresponds to the object in terms of integration. The matrix X introduced above is formed from the values $x_{i,j}$. These values represent three possibilities for each object: an object can belong to the class, can not belong and we can have no information at our disposal about the class membership of this object. Hence, the concept of object's class membership is the event/measurement in terms of statistics. At that, the classes to be integrated correspond to the categories and are the manifest variables that can be directly observed. The factor (latent class) is real–world class.

Further, class membership table introduced above corresponds to the Contingency Table 5.1. Data in this table is the input data for LCA and it represents either observed frequencies of patterns or p–values. Below we summarize p–values in the integration's terms:

p_i	–	observed proportion of objects assigned to the ith class ($i = 1, 2, \ldots, k$);
p_{ij}	–	observed proportion of objects assigned to both the ith and jth classes ($i \neq j$, $p_{ij} = p_{ji}$);
$p_{ij\ldots k}$	–	observed proportion of objects assigned to the ith, jth, …, kth classes ($i \neq j \neq \ldots \neq k$).

Values π_s and ν_{is} are the parameters which are to be found. Their meaning in the terms of integration is as following: $\pi_{s=1}$ value is the unobserved probability that a class was defined correctly, i.e., it corresponds to a real–world class. Value $\pi_{s=2}$ is the unobserved probability that a class was defined incorrectly, i.e., it is not a real–world class. Value ν_{i1} is the unobserved conditional probability that an object from the real–world class is also in the ith class. Value ν_{i2} is the unobserved conditional probability that an object, which is not from the real–world class is in the ith class.

Thus, we have equated the concepts of the integration task and the LCA. Further on we substantiate the developed framework with an illustrative example where we implement the statistical method for evaluating the integration process.

Suppose we have two data sources, which need to be integrated and a database resulted from their integration, i.e., integrated database. In Section 2.1 we provided three major steps for integrating the data sources with undefined structure and correspondences between the sources. These are: detecting dependencies and identifying semantics–carrying structure within each source, finding the correspondences between the sources on the schema and instance (data) level and applying results obtained on previous steps for schema and data integration. It is important to note, that these steps should be applied sequentially class by class, i.e., first the structure class–subclass in each source should be determined, then correspondences between classes from corresponding data sources should be revealed. At the data level, this means that first the objects belonging to each of the classes in each data source should be found and then matching objects from corresponding classes of corresponding data sources should be determined. As mentioned in Chapter 3, any of the methods, applied at each of these steps, never provide exact results and are always being accompanied, for example, by a certain level of confidence. The methods are applied to the schema on a class by class basis (they cannot handle the complete schema in one shot). Therefore, as a result of applying these methods each class is annotated with a value of confidence. Hence, for evaluating the complete integration process we need to include all these values into analysis, which means that we have to evaluate the integration process also class by class.

We can realize this principle if we apply the LCA to each triplet of the considered classes: two classes to be integrated and one integrated class, as shown in Figure 6.1.

Such representation of the task allows to estimate the result of the integration and also to estimate how correctly each of the source classes was determined and how correctly the correspondences between these classes were found. In the terms of the integration this means that we, using the LCA, intend to verify how close each of the classes in the triplet reflects real–world class. Obviously, if we get an answer for every class to be integrated, then we can evaluate the whole integration process.

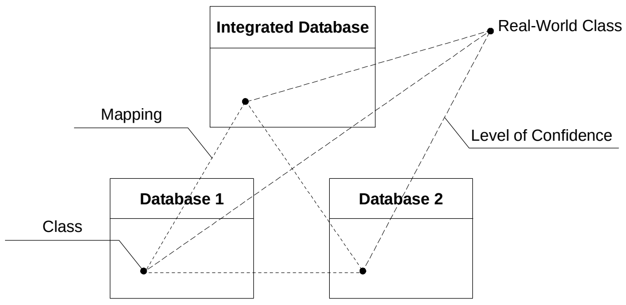

Figure 6.1: Triplet of classes to be considered in the LCA

Let us consider in detail the above case where two classes to be integrated (A_1 and A_2) and one integrated class A should be analyzed using the LCA. The general case of this integration is depicted in Figure 6.2.

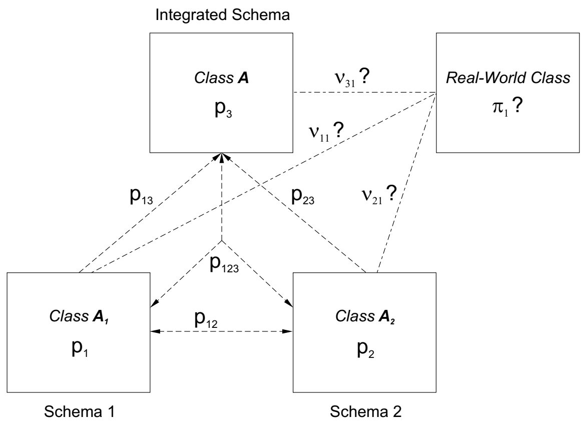

Figure 6.2: General case: evaluating the integration of two classes using LCA

As described in the paragraph *Construction of the class membership table p*–values, which are the input for the LCA can be derived from the methods applied during the integration process. Values p_1, p_2, and p_3 show the proportion of the objects that belong to the classes A_1, A_2, and A_3 respectively. Values p_{12}, p_{13}, and p_{23} represent the proportion of the objects that belong to each pair of classes A_1 and A_2, A_1 and A_3, and

A_2 and A_3 respectively. Value p_{123} is the proportion of objects that belongs to all three classes A_1, A_2, and A_3.

The LCA allows us to estimate the probability (π_1) with which all three considered classes correspond to the real–world class. Besides, it estimates probabilities with which every class (A_1, A_2, and A_3) corresponds to the real–world class.

The case considered above supposes integration of two data sources, that is why according to the LCA's identifiability condition (5.8), described in Section 5.2 we are forced to include the integrated database in the analysis. That answers the minimal case when the number of the latent classes is equal to 2 and number of categories is equal to 3. At that, it is necessary to analyze the integrated database in order to derive the values of p_3, p_{13}, p_{23}, and p_{123}. Besides, it is important that a set of objects considered in the analysis is a sample in the terms of statistics, and hence the number of the considered objects should be the same for all three data sources. This means, that if in the process of integration, for example, experts made decision not to include certain objects into the integrated database, then those objects should be taken out of the analysis completely. That way, even in case of integration of only two databases the integration result can be analyzed, but the real advantage of the method is revealed when the number of data sources to be integrated is three or more. That is because we can avoid including integrated database into analysis and evaluate integration results based on the initial data sources.

In the next section we demonstrate on examples how the statistical analysis evaluates various uncertainties.

6.2 Evaluation of the Integration Process Using LCA

6.2.1 Exact Solution for three Variables

In previous section we described the integration task in the terms of statistics. Now we would like to clarify on the numerical examples how integration can be evaluated. For that let us consider a simple case described in the previous section and depicted in Figure 6.2, where we have two classes (A_1 and A_2) from different data sources to be integrated and one integrated class (A) in the integrated database.

Thus, in this example we have the number of categories $k = 3$. The minimal number of latent classes $m = 2$ reserves for the objects only two possibilities: to be a member of the real–world class or not to be.

In order to demonstrate the LCA's working principle let us first consider an ideal case. Given is the following class membership 108×3 matrix X, which does not contain any missing data, i.e., it represents a case of class equivalence.

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 1 \\ \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Thus, the matrix X consists only of the definite part (100 rows with a pattern "1–1–1") and of the ideally random part (last 8 rows). The matrix being built by the last 8 rows is completely random. It contains 8 possible response patterns in equal proportions. The randomness of this part can be easily proved by calculating the correlation coefficients $r_{i,j}$ for every pair of categories (columns).

$$r_{i,j} = \frac{Cov(x_i, x_j)}{\sigma_i \sigma_j},$$

where covariance

$$Cov(x_i, x_j) = \frac{1}{n-1} \sum_{p=1}^{n} \left(x_{pi} - \frac{1}{n} \sum_{q=1}^{n} x_{qi} \right) \left(x_{pj} - \frac{1}{n} \sum_{q=1}^{n} x_{qj} \right)$$

and dispersion

$$\sigma_i = \sqrt{\frac{1}{n-1} \sum_{p=1}^{n} \left(x_{pi} - \frac{1}{n} \sum_{q=1}^{n} x_{qi} \right)^2}.$$

Hence, the correlation matrix $R = \{r_{i,j}\}$ for the random part shows no correlation between categories

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In this ideal case an exact solution of the system of equations (5.4) – (5.7) can be found, whereas in real applications such an ideal separation between definite and random content of the sample data does not take place. Therefore, in the real case, the random part has to be extracted using the best fit to the ideal one. Such a fitting procedure lays in the basement of the LCA.

In our ideal example represented by the matrix X the observed respond frequencies are given in the Table 6.3.

Table 6.3: Contingency table for the ideal case

Observed Pattern	**Number of occurrences**	**Observed frequency**
$1-1-1$	101	0.935
$1-1-0$	1	0.009
$1-0-1$	1	0.009
$1-0-0$	1	0.009
$0-1-1$	1	0.009
$0-1-0$	1	0.009
$0-0-1$	1	0.009
$0-0-0$	1	0.009

Using (5.11) we can find p–values:

$$\begin{aligned} p_1 &= \frac{104}{108} \\ p_2 &= \frac{104}{108} \\ p_3 &= \frac{104}{108} \\ p_{12} &= \frac{102}{108} \\ p_{13} &= \frac{102}{108} \\ p_{23} &= \frac{102}{108} \\ p_{123} &= \frac{101}{108} \end{aligned}$$

Substituting these p–values into the left hand side of the equations (5.4) – (5.7) yields two possible solutions:

$\pi_1 = \frac{25}{27}$	–	the unobserved probability of real–world class membership;
$\nu_{i1} = 1$	–	the unobserved conditional probability that an object from the real–world class (definite part) is correctly assigned to the ith data source;
$\pi_2 = \frac{2}{27}$	–	the unobserved probability of real–world class non–membership;
$\nu_{i2} = \frac{1}{2}$	–	the unobserved conditional probability that an object which is not in real–world class (random part) is incorrectly assigned to the ith data source.

and

$\pi_1 = \frac{2}{27}$	–	the unobserved probability of real–world class membership;
$\nu_{i1} = \frac{1}{2}$	–	the unobserved conditional probability that an object from the real–world class (definite part) is correctly assigned to the ith data source;
$\pi_2 = \frac{25}{27}$	–	the unobserved probability of real–world class non–membership;
$\nu_{i2} = 1$	–	the unobserved conditional probability that an object which is not in real–world class (random part) is incorrectly assigned to the ith data source.

These solutions are symmetric, i.e., one can be turned into the other and vise versa by swapping their response category indices and cross naming the object groups "latent class membership" and "latent class non–membership". A choice between these two solutions is actually a choice of response category that is considered as a real–world class. It is naturally based on the initial assignment of object to classes, i.e., response "1" is counted in favor of real–world class. Most of objects in the sample usually respond positively to class membership in every of three data sources. In our case the first solution has to be chosen.

The second solution reflects a simple and obvious fact that alternative initial assigning of object to classes, i.e., when response "0" is counted in favor of real–world class naturally leads to a low presence of this class in the sample.

One can imagine a situation where the scores of "1" and "0" in the contingency table are almost equal. In this case a correct choice of appropriate solution is not possible.

Thus, the LCA gives us an answer to the question what proportion of objects belongs to the real–world class ($\pi_1 = \frac{25}{27} \approx 0.926$). Indeed, this value demonstrates a confidence with which the integrated class corresponds to real–world class.

We see that $\frac{2}{27}$ of objects ($\approx 7.4\%$) do not belong to real–world class. They have equal probability (50%) to be found or not to be found in each data set. This follows from the initial assumptions about the statistical nature of our uncertainties.

On this example we can see that the estimation value π_1 of the real–world class is different from p_3. That is a consequence of the fact that by using only the p_3–value we cannot make a certain conclusion about how integrated class corresponds to the real–world class, because the p_3–value represents the support of class membership for an integrated class and does not take into account the confidence value for this class, which depends on several values of the class correspondence between the data sources.

On contrary, the π_1–value takes into account all these uncertainties and gives us an integrated value of support of real–world class membership for the integrated set of objects.

6.2.2 An Example of Applying LCA

In this section we return to the example considered in Section 4.3. In this example we studied the problems, appearing with an attempt to estimate the result of the integration of the data, that contains uncertainties. To be more concrete, we present this example according to Figure 4.6, where the abstract classes A_1, A_2, B_1, B_2 and so on are substituted with concrete classes *All Clients*, *All Customers*, *Late on their payment*, *Debtors*, etc. respectively.

Below we consider two cases of evaluation of the integration shown in Figure 4.6, namely a case of class equivalence and a case of class overlapping. The equivalence case corresponds to a situation where the same set of clients is represented differently in two existing data sources. In the case where classes of different data sources overlap, there are clients, that are present in two sources at the same time and additionally, each data source contains a number of clients, that are missing in the other data source. As a result of applying data mining, schema matching, etc. methods, both sources have been analyzed on a matter of belonging of objects to classes and on the correspondence between classes/objects. Further, we suppose, that as a result of analysis and integration steps, a set of objects has been selected, that enters the integrated schema, and for that set correspondences and support values have also been determined. The obtained values agree with those shown in Figure 4.6.

Equivalence case. Let us consider a case of equivalence, where all of the objects, selected for integration, belong to each data source. In this example (Figure 4.6) the integrated schema copies the source schema 1, therefore it can be naturally assumed, that the p–values for the integrated schema have the same values as those in schema 1, i.e., $p_3 = p_1$, $p_{13} = p_1$, $p_{23} = p_{12}$, and $p_{123} = p_{12}$, where index 3 refers to the integrated

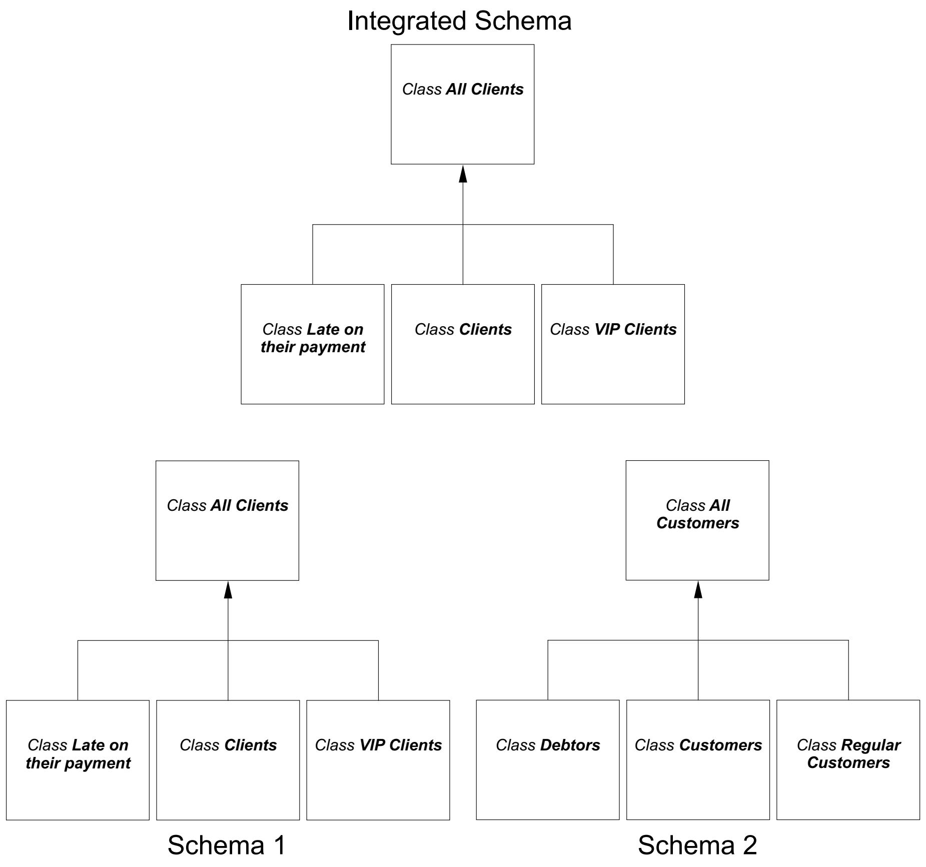

Figure 6.3: Integration of two data sources

schema. We should emphasize that if we make this assumption, then the LCA cannot be correctly applied, because the complete correlation between two data sources would surface in the LCA as an influence of the latent (we assume real–world) factor. This could immediately be seen in the results of the calculations, carried out under these assumptions. We demonstrate this considering following classes: class *Late on their payment*, its corresponding class *Debtors* and integrated class *Late on their payment*. Taking the numbers from Figure 4.6 we get the following set of values p:

$$\begin{aligned} p_1 &= 0.87 \\ p_2 &= 0.84 \\ p_3 &= 0.87 \\ p_{12} &= 0.80 \\ p_{13} &= 0.87 \\ p_{23} &= 0.80 \\ p_{123} &= 0.80 \end{aligned}$$

and a Contingency Table 6.4. This table is used as input data by the program WinLTA that realizes the maximum likelihood estimation method.

Table 6.4: Option 1: contingency table for the equivalence case, 1000 objects

Observed Pattern	**Number of occurrences**	**Observed frequency**
$1-1-1$	800	0.8
$1-1-0$	0	0
$1-0-1$	70	0.07
$1-0-0$	0	0
$0-1-1$	0	0
$0-1-0$	40	0.04
$0-0-1$	0	0
$0-0-0$	90	0.09

The results of the application of the LCA look like as follows:

$\pi_1 = 0.87$	–	the unobserved probability of real–world class membership;
$\nu_{11} = 1$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the class *Late on their payment*;
$\nu_{21} = 0.92$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the class *Debtors*;
$\nu_{31} = 1$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the integrated class *Late on their payment*;
$\pi_2 = 0.13$	–	the unobserved probability of real–world class non–membership;
$\nu_{12} = 0$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the class *Late on their payment*;
$\nu_{22} = 0.008$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the class *Debtors*;
$\nu_{32} = 0$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the integrated class *Late on their payment*.

It is worth to note, that the goodness of fit for this calculation is close to zero. It indicates an exact fit of the statistically expected frequencies to the input ones.

These results show that the equivalence between the first and the third (integrated) schema, which we introduced, led us to an incorrect result. Namely, low probabilities of class correspondences between the integrated data sources had no influence on π_1 compared with p_1, which does not conform to our interpretation. Hence, we demonstrated unconditional importance of the independent analysis of each of the three data sources, included in the LCA task. Therefore, we cannot yet answer the questions stated in Section 4.3 Figure 4.6. These answers could be given if we include into consideration at least one more independent data source or if we analyze the integrated classes in order to obtain independent p–values. If the objects are introduced into the integrated database exclusively by two initial sources, then independent analysis of the integration result is not possible. If we assume a broader notion of integration, namely, suppose, that candidates for the membership in the integrated database do exist and we select them using our data sources, then such analysis is possible. For example, we have a list of the potential clients of our firm (in general case list of all the city residents) and we would like to determine possible real clients using data of two other similar firms, operating in the city some time ago. Obviously, changes, occurred with the flow of time introduce uncertainties into the problem of relating the real clients (people living currently) to the records about the clients, kept in the old data sources. Then we have to determine correspondences between the objects in our list with the objects of the two sources and according to the result distribute them into the subclasses *Late on their payment*, *Clients* or *VIP Clients*. It is obvious that such an analysis represents an independent analysis of the integrated schema, giving as a result all needed p–values.

Now we continue the analysis suggesting that these values are available, that is we add some arbitrary p–values, related to the integrated class *Late on their payment*. Then as p–values data we should have something like:

$$\begin{aligned} p_1 &= 0.87 \\ p_2 &= 0.84 \\ p_3 &= 0.88 \\ p_{12} &= 0.80 \\ p_{13} &= 0.81 \\ p_{23} &= 0.81 \\ p_{123} &= 0.78 \end{aligned}$$

Table 6.5 is the corresponding contingency table.

Table 6.5: Option 2: contingency table for the equivalence case, 1000 objects

Observed Pattern	**Number of occurrences**	**Observed frequency**
$1-1-1$	780	0.78
$1-1-0$	20	0.02
$1-0-1$	30	0.03
$1-0-0$	40	0.04
$0-1-1$	30	0.03
$0-1-0$	10	0.01
$0-0-1$	40	0.04
$0-0-0$	50	0.05

The results of the LCA's application look like as follows:

$\pi_1 = 0.81$	–	the unobserved probability of real–world class membership;
$\nu_{11} = 0.972$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the class *Late on their payment*;
$\nu_{21} = 1$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the class *Debtors*;
$\nu_{31} = 0.984$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the integrated class *Late on their payment*;
$\pi_2 = 0.19$	–	the unobserved probability of real–world class non–membership;
$\nu_{12} = 0.437$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the class *Late on their payment*;
$\nu_{22} = 0.160$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the class *Debtors*;
$\nu_{32} = 0.437$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the integrated class *Late on their payment*.

Goodness of fit is $G^2 = 0.042$, which is an acceptable value. In real application it can be expected that the solution does not as easily converge, then the parameters of the latent model should be changed until the goodness of fit test value is acceptable.

To a high degree this solution resembles the result for the ideal case, considered in the beginning of the section. It gives a realistic illustration of how the different kinds of uncertainties can be converged to a single π_1–value under the condition, that the initial uncertainties have statistical nature and are correctly defined.

In a similar manner the calculations for all the remaining subclasses of the lowest level of the integrated schema can be carried out, in our case classes *Clients* and *VIP Clients*. In order to evaluate the superclasses *All Clients*, *All Customers* and integrated class *All Clients* the analysis using the LCA should not be repeated, since they, as well as all the classes of the highest level, are the simple unions of the set of objects of the lowest level. Therefore, for determining the π_1–value for a superclass, an average π_1–value, weighted by all the subclasses could be taken. Let the total number of objects be N, the number of objects in subclasses *Late on their payment*, *Clients* and *VIP Clients* be n_1, n_2 and n_3 respectively.

We denote π_1–values delivered by the LCA for each of the subclasses as π_1^1, π_1^2, and π_1^3 respectively. Then the probability π_1 that the superclass *All Clients* in the integrated schema corresponds to the real–world class could be calculated according to the following formula:

$$\pi_1 = \frac{\pi_1^1 n_1 + \pi_1^2 n_2 + \pi_1^3 n_3}{N}. \tag{6.1}$$

Considering that $N = \sum_{i=1}^{k} n_i$ formula 6.1 in the general case takes the following form:

$$\pi_1 = \frac{1}{N} \sum_{i=1}^{k} \pi_1^i n_i, \tag{6.2}$$

where $i = 1, 2, \ldots k$ refers to the corresponding subclass.

Overlapping case. Now we broaden the analysis of our last example of integration of class *Late on their payment* and its corresponding class *Debtors* into the integrated class *Late on their payment* by setting that they overlap. We set, that additionally to the 1000 equivalent objects, class *Late on their payment* contains 400 more objects that are missing in class *Debtors*, and class *Debtors* contains 300 more objects that are missing in the class *Late on their payment*.

Let m_i be an number of missing objects in ith category, then the corresponding frequencies and a number of corresponding respond patterns when the data is missing in the 1st and in the 2nd categories (denoted as "2" in the respond pattern) is determined as shown in Table 6.6:

The variables p_1, p_2, p_3, p_{13}, p_{23} are known. The frequencies f_{i2k} and f_{2jk}, which correspond only to the respective set of missing objects, are unknown. We should suggest for these sets the same p–values as for the complete set of objects. Then the following set of equations results, having missing data in the 1st category:

Table 6.6: Number of occurrences of respond patterns with missing data

Observed Pattern	**Number of occurrences**	**Observed frequency**
$2-1-1$	$f_{211} \times m_1$	f_{211}
$2-0-1$	$f_{201} \times m_1$	f_{201}
$2-1-0$	$f_{210} \times m_1$	f_{210}
$2-0-0$	$f_{200} \times m_1$	f_{200}
$1-2-1$	$f_{121} \times m_2$	f_{121}
$0-2-1$	$f_{021} \times m_2$	f_{021}
$1-2-0$	$f_{120} \times m_2$	f_{120}
$0-2-0$	$f_{020} \times m_2$	f_{020}

$$\begin{aligned} p_2 &= f_{211} + f_{210} \\ p_3 &= f_{211} + f_{201} \\ p_{23} &= f_{211} \end{aligned}$$

In case that there is missing data in the 2nd category, the set of equations looks like as follows:

$$\begin{aligned} p_1 &= f_{121} + f_{120} \\ p_3 &= f_{121} + f_{021} \\ p_{13} &= f_{121} \end{aligned}$$

That gives:

$$\begin{aligned} f_{211} &= p_{23} \\ f_{201} &= p_3 - p_{23} \\ f_{210} &= p_2 - p_{23} \\ f_{200} &= 1 + p_{23} - p_3 - p_2 \\ f_{121} &= p_{13} \\ f_{021} &= p_3 - p_{13} \\ f_{120} &= p_1 - p_{13} \\ f_{020} &= 1 + p_{13} - p_3 - p_1 \end{aligned}$$

In our example $m_1 = 300$ and $m_2 = 400$. The computation outcome is shown in Table 6.7.

Table 6.7: Contingency table for the overlapping case, 1700 objects

Observed Pattern	**Number of occurrences**	**Observed frequency**
$1-1-1$	780	0.459
$1-1-0$	20	0.012
$1-0-1$	30	0.018
$1-0-0$	40	0.040
$0-1-1$	30	0.018
$0-1-0$	10	0.006
$0-0-1$	40	0.024
$0-0-0$	50	0.029
$2-1-1$	243	0.810
$2-0-1$	21	0.070
$2-1-0$	9	0.030
$2-0-0$	27	0.090
$1-2-1$	324	0.810
$0-2-1$	28	0.070
$1-2-0$	24	0.060
$0-2-0$	24	0.060

The results of the LCA's application look like as follows:

$\pi_1 = 0.81$	–	the unobserved probability of real–world class membership;
$\nu_{11} = 0.972$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the class *Late on their payment*;
$\nu_{21} = 1$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the class *Debtors*;
$\nu_{31} = 0.984$	–	the unobserved conditional probability that the objects from the real–world class are correctly assigned to the integrated class *Late on their payment*;

$\pi_2 = 0.19$	–	the unobserved probability of real–world class non–membership;
$\nu_{12} = 0.438$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the class *Late on their payment*;
$\nu_{22} = 0.159$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the class *Debtors*;
$\nu_{32} = 0.438$	–	the unobserved conditional probability that the objects which are not in real–world class are incorrectly assigned to the integrated class *Late on their payment*.

These results fully conform to the results obtained for the case of equivalence since here we used the same input p–values. This demonstrates the ability of the method to handle overlapping classes as easy as equivalent ones. Thus, accounting for the overlapping classes does not require a special analysis. The case with overlapping classes can naturally be included into consideration.

6.3 Integration Quality (IQ)

In this section we review a problem of the integration quality (IQ). Under IQ we understand correspondence of obtained integrated database to the real–world database. The contribution of our work consists of proposing the method, capable of estimating the integration process.

As has been repeatedly mentioned above, during the integration of the heterogeneous data sources a number of conflicts and discrepancies appears that could be resolved by various methods that introduce uncertainties into the integration process. Application of the LCA by no means supposes resolution of the part of the arising conflicts or improvement of the work of some methods, but rather only allows to account for the uncertainties of different nature and as result gives common estimation to the whole integration process, i.e., correspondence of the derived result to the real–world.

LCA's working principle for estimating the IQ is following. Consider integration developing according to the bottom–up scenario, i.e., the lowest level of each integration data source is a determining basic object level, from which other levels are derived. The real–world class membership (π_1), calculated by the LCA methods, is delivered for this lowest database level. The other real–world class memberships for the classes along the bottom–up hierarchy are the weighted average of their parent nodes. Exercising this principle all the way up, objects membership for the whole tree is calculated.

As was shown in the examples of the previous section, the input data for the LCA are the support values for each of the considered node (class) membership and also correspondence values between these nodes. Therefore, for evaluating real–world class membership at the lowest level, these values should be calculated, at that, during the calculation, only those classes/objects should be studied, which afterwards will be included in the integrated database, i.e., by the time that the LCA is activated, all the preceding integration work has to be accomplished.

During the estimation of the IQ by the LCA method, it is also important that according to the applicability condition at least three data sources should be provided. Therefore, during the evaluation of the integration of two data sources, the integrated database should be involved in the analysis as well, as was shown in the examples of the previous section. In the integration case of three and more data sources the integrated database should not be included in the analysis, and hence, there is no need to analyze the integrated database with the purpose of discovering the membership support values as well as the class correspondence values. Conclusions about the integrated database could be drawn only based on the weighted average π_1–values, derived for the sources to be integrated.

The proposed approach allows to improve the IQ using an iteration principle as shown in Figure 6.4.

Thus, as could be seen from this figure, the IQ improvement procedure comes to estimating the integration process by the method of the LCA, and then, if the result's quality is unacceptable, the previous steps of the integration process can be repeated with new input data, i.e., using alternative methods/strategies for: structure extraction, finding correspondences, integration, and as a consequence different resulting sample of objects is considered in the LCA. Besides that, the input parameters in used methods can be changed, for example, for schema matching methods a new learning set can be built or another dictionary can be used, etc. The cycle should continue until the acceptable quality is attained.

It is important to underscore, that the LCA is intended for the integral estimate and accounting for uncertainties based on the already selected set of objects, hence LCA cannot give any recommendation as to which exactly objects should be selected and how exactly the parameters of the applied methods should be altered. Similarly it cannot find a more effective method for solving some of the integration tasks. In this case experts' experience proves to be decisive, since the integration of the heterogeneous data sources is an ambiguous process, and depending on the given data and the area of application, decisions can vary a lot. It is easy to imagine such a situation, where an attempt to determine classes more precisely, i.e., to increase the value of class membership support (p_i–values), leads to decreasing the values of class correspondence support (p_{ij}–values), if the objects across classes are compared according to the same strict rules. This

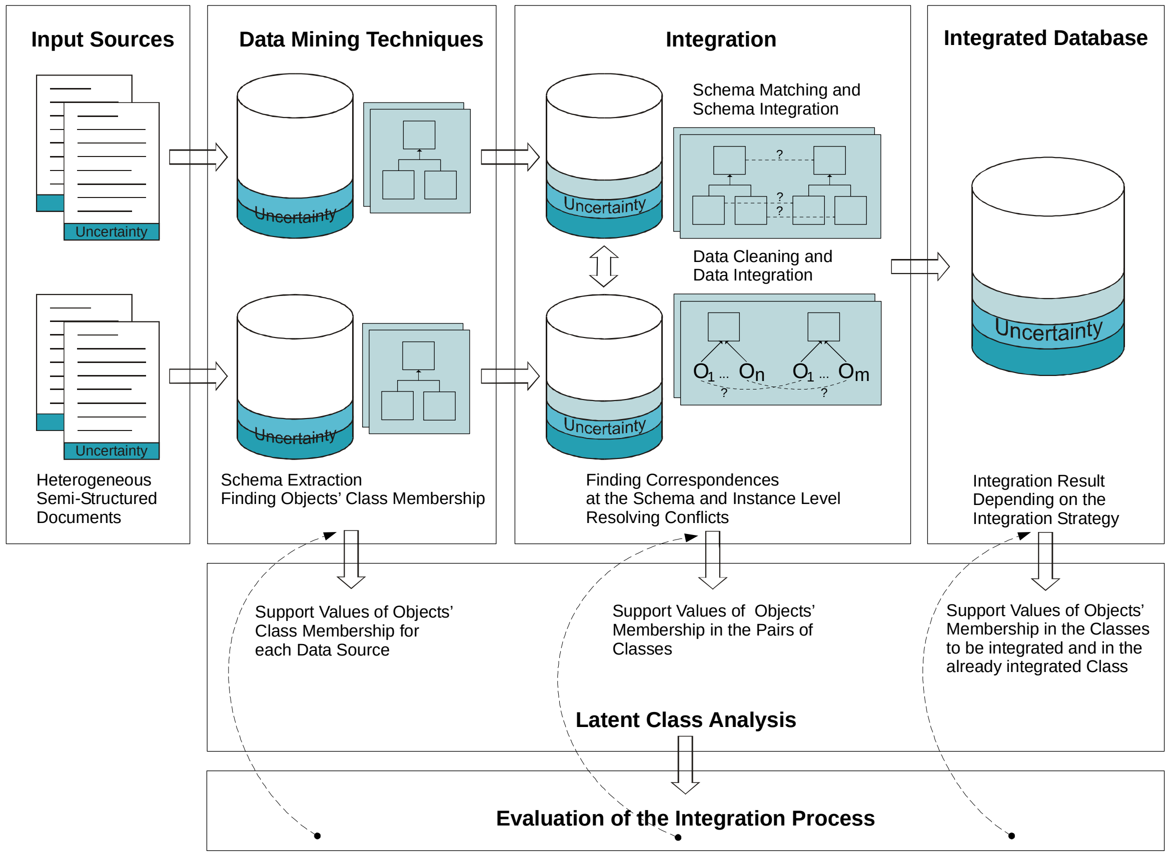
Input Sources
Uncertainty
Uncertainty
Heterogeneous Semi-Structured Documents
Data Mining Techniques
Uncertainty
Uncertainty
Schema Extraction
Finding Objects' Class Membership
Integration
Schema Matching and Schema Integration
Uncertainty
Data Cleaning and Data Integration
Uncertainty
O1 ... On
O1 ... Om
Finding Correspondences at the Schema and Instance Level
Resolving Conflicts
Integrated Database
Uncertainty
Integration Result Depending on the Integration Strategy
Support Values of Objects' Class Membership for each Data Source
Support Values of Objects' Membership in the Pairs of Classes
Support Values of Objects' Membership in the Classes to be integrated and in the already integrated Class
Latent Class Analysis
Evaluation of the Integration Process

Figure 6.4: Improvement of the integration quality

happens because the methods, determining p_i–values and p_{ij}–values work independently and cannot account for the mutual influence. Then using the LCA becomes important for determining the selection parameters. Computing the π–values does not present a problem when input data (p–values) is determined. Thus, the LCA's role becomes obvious at the stage where it is necessary to estimate the result of the combined influence of the different uncertainty types and to make a decision, in this way improving IQ.

6.4 Summary

In this chapter we have proposed to account for uncertainties in the integration process using the available statistical method LCA. For that in Section 6.1 we build statistical model of the integration task. Building the model is subdivided into the following steps: data representation in the integration task, constructing the class membership table, and formulating the integration task in terms of the statistical model.

Viewing classes as categories in the LCA and considering that p–values correspond to the support of the class membership and support of class correspondence, we can, using

LCA, get the π–values, which correspond to the real–world class memberships.

Since the minimal case that could be considered by the LCA method supposes presence of three categories, then for integrating only two data sources we have to include an integrated database into the analysis. Testing the triples of the corresponding classes on a real–world class membership, i.e., two classes to be integrated and one integrated class, we get the π–values, accounting for the combined influence of all types of the uncertainties.

It is important that for integrating three and more sources, we do not have to include the integrated database into the analysis. For testing if the whole integrated database corresponds to the real world we have to obtain π–values for each triplet at the lowest level and then taking weighted average we can move up the tree accordingly.

In Section 6.2 the examples of applying LCA for evaluating the integration process have been given. First, an ideal case with three categories, which contains only a definite and an ideally random part, has been considered. This example demonstrated that for integration of three data sources, the obtained π_1–value is different than the p_3–value, i.e., it was shown that the value π includes influence of uncertainties of class correspondence, whereas p–value does not and reflects only the support of class membership.

Another example also dealt with integration of two data sources. We made an attempt to demonstrate that replicating the integrated database from the initial database leads to the incorrect result, namely to the π–value being equal to the p–value of the copied class.

Then we considered a case of integration of two data sources, calculating p–values for the third database independently. The received result proves that π_1 considers influence of all the uncertainties on the integrated database. Besides that an example has been given, reviewing the case when the classes overlap and providing an acceptable result. Thus, LCA can estimate the case of class overlapping.

In Section 6.3 problems of quality of the integration process have been touched. We demonstrated that the proposed method can assist in improving integration quality using iterations. One could try out different integration strategies, use various techniques to derive schema information, schema correspondences, etc. Thereby changing objects entering the integrated database, i.e., providing new data as an input to LCA and calculating the result, one can repeat the procedure until the acceptable quality is attained.

7 Conclusion

This chapter presents a summary of the contributions of this thesis. Section 7.1 gives a statement of the problem and in Section 7.2 we describe our contributions to the solution of the given problem. Section 7.3 provides an outlook for our work.

7.1 Problem Statement

We consider integration process for heterogeneous semi–structured data or legacy systems.

Heterogeneity of these sources appears as a consequence of them being created in different time and different places by different people. Due to these reasons there is a chance that the same data is represented in different ways. Thus, sources that have to be integrated could use different data models and therefore during the integration a decision should be made about the type of model for the integrated schema. Besides that, the same real–world object could be presented by different structural elements in schemas. Additionally, conflicts, caused by homonyms and synonyms used at different schema levels (classes, attributes, objects) could complicate integration. And, of course, the most serious conflicts during the integration of the heterogeneous data sources are caused by the not completely known data source semantics.

Besides that, semi–structured sources and legacy systems do not always possess the complete schema, the semantics is not always completely known and correspondences between such sources are not known.

Therefore, integrating such data sources is not possible, using only the available integration methods, because they suppose complete and precise knowledge about sources and their relationships. We, hence, have to apply to our data sources such methods as data mining, schema matching, data cleaning, etc. These methods allow to extract the necessary information for integration. They, however, do not provide a reliable result, but rather a result with a certain trustworthiness. The appearing in this way uncertainties could distort the integration result in an unpredictable way.

For the integration task it is important to know how good the result reflects the reality. Integration methods proposed so far are not capable of accounting for existing uncertainties and accordingly are not able to evaluate the integration result, i.e., cannot provide an estimate of the integration quality.

The aim of our work is to develop an approach that is able to take into account all existing uncertainties and give a quantitative estimation to the integration quality.

7.2 Contributions

In this thesis we have considered the problem of integrating heterogeneous data sources from the point of view of possible integration quality estimation. For that we reviewed existing analysis methods of such data and analyzed the proposed possible integration process. The following stages of the process were introduced: Detecting dependencies and identifying semantics–carrying structure within each source, finding the correspondences between the sources on the schema and instance (data) level and applying results obtained at previous steps for schema and data integration. A crucial problem is that all the methods employed return their results with a limited confidence. Having focused on this problem we suggested classifying uncertainties by their types, namely: uncertainty about the exact structure of data objects, uncertainty concerning the assignment of data objects to classes, uncertainty concerning the extensional correspondence between classes from two data sources.

Most attention has been given to accounting for interaction between different types of uncertainties. We suggested an approach using the statistical analysis principles that account for the influence of the latent factors on the behavior of the data aggregate — Latent Class Analysis (LCA), which made it possible to interpret the correlation of discovering the identical objects in different data sources as an influence of such universal factor as belonging of objects to a respective real–world class.

We have reviewed theoretical framework of the LCA method and the conditions under which its application is possible and advisable. Special attention has been given to building the statistical model for the uncertainty analysis and to discovering the correspondences between model parameters and the data source parameters.

It has been especially emphasized that it is essential for the analysis to have at least three (or more) independent data sources, containing the set of objects suspected to be equivalent, normally associated with the certain classes in the data structure. In this case the analysis results into the probability value of the real–world class membership for a given group of objects. In case where only two initial schemas are being integrated, the role of the compulsory third source could be played by the integrated schema, where an independent determination of all support and confidence parameters involved in the statistical analysis is provided. In other words, these parameters should not be derived from the initial data sources.

The value of the real–world class membership returned by the LCA is an integral parameter, which combines the influences of all types of uncertainties. That is why it reflects the quality of integration. If we define this value for all groups of objects that form the subclasses of the lowest schema level, then for any class of any level, including the whole database, we get the support of real–world class membership as a simple weighted average of these values over included subclasses.

It is important to stress that the given method does not solve the common problem of integration of the heterogeneous data sources. It represent a useful instrument for evaluating the integration results, which under certain conditions can be used for improving quality of selection of the data, containing uncertainties.

7.3 Outlook

In this work we have reviewed the basic principles that can be employed and the conditions that have to be satisfied for applying statistical analysis to the integration of several heterogeneous and semi–structured data sources. We supposed that the data required for this analysis can be derived by such methods as data mining, schema matching, data cleaning, etc., and that this data corresponds unambiguously to the p–values, used by the statistical analysis. In practice, there might be the cases that these assumptions are violated, i.e., some of the p–values could be lacking. This is especially possible when we have to analyze the integration of more than three data sources. In Section 5.2 we have defined the necessary condition for LCA's identifiability (see equation 5.8). In case that $m = 2$ and $k > 3$, not all the equations are mandatory for obtaining a solution. This means that it is not necessary to use the joint probabilities higher than a few first orders, i.e., p_i, p_{ij}. In statistics a number of different models have been developed, depending on the type and amount of information employed (i.e., depending on the number of variables) and the number of joint probabilities, which are used. A possible direction of future work can be investigating applicability of LCA models capable of solving system with the incomplete set of equations, i.e., when not all the joint probabilities are given.

Another problem requiring more detailed analysis is building the mathematical model, in which the p–values are not independent (Restricted Model). Such breach of independence can be caused by additional conditions and limitations emerging as a result of obtaining the support and confidence values (p–values) by the data source analysis methods.

In the thesis we demonstrated how our method could be applied to evaluate integration process and improve IQ on some small integration examples. Large scale application crosses the scope of one dissertation, because the integration process of legacy or semi–structured data sources consists of many steps (defining structure and correspondences, resolving conflicts, etc.), the process is complicated and ambiguous. It would be nevertheless interesting to review a large scale integration example in the future works.

List of Figures

Bibliography

[AC01] E. Altareva and S. Conrad. The Problem of Uncertainty and Database Integration. In R.-D. Kutsche, S. Conrad, and W. Hasselbring, editors, *Engineering Federated Information Systems, Proceedings of the 4th Workshop EFIS 2001, Oct 9–10, 2001, Berlin (Germany)*, pages 92–99. infix–Verlag / IOS Press, 2001.

[AC03a] E. Altareva and S. Conrad. Analyzing Uncertainties in the Database Integration Process by Means of Latent Class Analysis. In *Engineering Federated Information Systems, Proceedings of the 5th Workshop EFIS 2003, July 17–18, 2003, Coventry (UK)*, pages 14–24. Akad. Verlagsgem / IOS Press (infix), 2003.

[AC03b] E. Altareva and S. Conrad. Statistical Analysis as Methodological Framework for Data(base) Integration. In Il-Yeol Song, Stephen W. Liddle, Tok Wang Ling, and Peter Scheuermann, editors, *22nd International Conference on Conceptual Modeling – ER 2003, Chicago, IL, USA, October 2003*, pages 17–31. Springer–Verlag, LNCS, 2003.

[AC03c] E. Altareva and S. Conrad. Unsichere Informationen bei der Datenbankintegration und ihre Behandlung im Integrationsprozess. *Datenbank–Spektrum*, 6:14–22, 2003.

[Bas94] A. Basilevsky. *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley and Sons, New York, 1994.

[BBGV01] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. The MOMIS Approach to Information Integration. In *ICEIS 2001, Proceedings of the 3rd International Conference on Enterprise Information Systems, Setubal, Portugal, July 7–10, 2001*, 2001.

[BCV99] S. Bergamaschi, S. Castano, and M. Vincini. Semantic Integration of Semi-structured and Structured Data Sources. *SIGMOD Record*, 28(1):54–59, 1999.

[BFN94] R. Busse, P. Fankhauser, and E. J. Neuhold. Federated Schemata in ODMG. In J. Eder and L. A. Kalinichenko, editors, *East/West Database Workshop: Klagenfurt 1994, Workshops in Computing*, pages 356–379. Springer–Verlag, 1994.

[BG01] A. Bauer and H. Günzel, editors. *Data–Warehouse–Systeme*. dpunkt–Verlag, Heidelberg, 2001.

[BHP92] M.W. Bright, A.R. Hurson, and S. H. Pakzad. A Taxonomy and Current Issues in Multidatabase Systems. *IEEE Computer Society*, 25(3):50–60, 1992.

[BK99] D. Bartholomew and M. Knott. *Latent Variable Models and Factor Analysis*, volume 7 of *Kendall's Library of Statistics*. Arnold, London, 1999.

[BLN86] C. Batini, M. Lenzerini, and S.B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[BM99] C. Beeri and T. Milo. Schemas for Integration and Translation of Structured and Semi–structured Data. In C. Beeri and P. Buneman, editors, *Database Theory — ICDT '99, 7th International Conference, Jerusalem, Israel, January 10–12, 1999, Proceedings*, Lecture Notes in Computer Science, Vol. 1540, pages 296–313. Springer–Verlag, 1999.

[BP85] D. P. Ballou and H. L. Pazer. Modeling Data and Process Quality in Multi–Input, Multi–Output Information Systems. *Management Science*, 31(2):150–162, 1985.

[CFOA02] S. Castano, A. Ferrara, G. S. K. Ottathycal, and V. De Antonellis. A Disciplined Approach for the Integration of Heterogeneous XML Datasources. In *13th International Workshop on Database and Expert Systems Applications (DEXA 2002), 2–6 September 2002, Aix-en-Provence, France*, pages 103–110. IEEE Computer Society, 2002.

[CHJ$^+$96] S. Conrad, M. Höding, S. Janssen, G. Saake, I. Schmitt, and C. Türker. *Integrity Constraints in Federated Database Design*. Preprint Nr. 2, Fakultät für Informatik, Universität Magdeburg, 1996.

[CHY96] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from a Database Perspective. *IEEE Transactions on Software Engineering*, 8(6):866–883, 1996.

[Con97] S. Conrad. *Föderierte Datenbanksysteme: Konzepte der Datenintegration.* Springer–Verlag, Berlin/Heidelberg, 1997.

[DDL00a] A. Doan, P. Domingos, and A. Y. Levy. Learning Mappings between Data Schemas. In *Proceedings of the AAAI–2000 Workshop on Learning Statistical Models from Relational Data, 2000, Austin, TX*, 2000.

[DDL00b] A. Doan, P. Domingos, and A. Y. Levy. Learning Source Description for Data Integration. In D. Suciu and G. Vossen, editors, *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18–19, 2000*, pages 81–86, 2000.

[DJ03] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning.* John Wiley & Sons, Hoboken, New Jersey, USA, 2003.

[DM92] W. H. Delone and E. R. McLean. Information Systems Success: the Quest for the Dependent Variable. *Information Systems Research*, 3(1):60–95, 1992.

[DMR03] H. H. Do, S. Melnik, and E. Rahm. Comparison of Schema Matching Evaluations. In A. B. Chaudhri, M. Jeckle, E. Rahm, and R. Unland, editors, *Web, Web–Services, and Database Systems, NODe 2002 Web and Database–Related Workshops, Erfurt, Germany, October 7–10, 2002*, volume 2593 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2003.

[FLMC01] W. Fan, H. Lu, S. E. Madnick, and D. W.-L. Cheung. Discovering and Reconciling Value Conflicts for Numerical Data Integration. *Information Systems*, 26(8):635–656, 2001.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(4):37–54, 1996.

[GFS$^+$01] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative Data Cleaning: Language, Model and Algorithms. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *Proceedings of the 27th International Conference on Very Large Databases (VLDB), September 11–14, Rome,Italy, 2001*, pages 371–380. Morgan Kaufmann, 2001.

[Goo74] L. A. Goodman. *Exploratory Latent Structure Analysis Using both Identifiable and Unidentifiable Models.* Biometrika, 1974.

[Goo95] D. L. Goodhue. Understanding User Evaluations of Information Systems. *Management Science*, 41(12):1827–1844, 1995.

[GSSC95] M. García-Solaco, F. Saltor, and M. Castellanos. A Structure Based Schema Integration Methodology. In P. S. Yu and A. L. P. Chen, editors, *Proceedings of the 11th International Conference on Data Engineering, March 6–10, 1995, Taipei, Taiwan*, pages 505–512. IEEE Computer Society, 1995.

[GSW02] H. Graubitz, M. Spiliopoulou, and K. Winkler. The DIAsDEM Framework for Converting Domain–Specific Texts into XML Documents with Data Mining Techniques. In *Proceedings First IEEE International Conference on Data Mining, San Jose, CA, USA*, pages 171–178, 2002.

[GWS01] H. Graubitz, K. Winkler, and M. Spiliopoulou. Semantic Tagging of Domain–specific Text Documents with DIAsDEM. In K.-U. Sattler, editor, *Proceedings 1st International Workshop on Databases, Documents, and Information Fusion (DBFusion 2001), Magdeburg, Germany, May 2001*, pages 61–72, 2001.

[HC98] M. Höding and S. Conrad. Data–Mining Tasks in Federated Database Systems Design. In T. Özsu, A. Dogac, and Ö. Ulusoy, editors, *Issues and Applications of Database Technology (IADT'98), Proceedings of the 3rd World Conference on Integrated Design and Process Technology, July 6–9, 1998, Berlin, Germany*, volume 2, pages 384–391. Society for Design and Process Science, 1998.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, San Francisco, CA, USA, 2001.

[HM85] D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, 1985.

[HMS01] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, Massachusetts Institute of Technology, USA, 2001.

[IJG03] R. Iqbal, A. E. James, and R. A. Gatward. A Framework for Integration of Heterogeneous Systems. In *New Technologies for Information Systems, Proceedings of the 3rd International Workshop on New Developments in Digital*

Libraries, NDDL 2003, and the 1st International Workshop on Validation and Verification of Software for Enterprise Information Systems, VVEIS 2003, In conjunction with ICEIS 2003, Angers, France, April 2003, pages 56–65. ICEIS Press, 2003.

[KCGS] W. Kim, I. Choi, S. Gala, and M. Scheevel. On Resolving Schematic Heterogeneity in Multidatabase Systems. pages 521–550.

[Kim96] R. Kimball. Dealing with Dirty Data. *DBMS Magazine*, 9(10), 1996.

[KS91] W. Kim and J. Seo. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer*, 24(12):12–18, 1991.

[Leh03] W. Lehner. *Datenbanktechnologie für Data–Warehouse–Systeme: Konzepte und Methoden*. dpunkt.verlag, Heidelberg, 2003.

[LH68] P. F. Lazarsfeld and N. W. Henry. *Latent structure analysis*. Houghton Mifflin, Boston, 1968.

[LLL01] W. L. Low, M.-L. Lee, and T. W. Ling. A knowledge–based Approach for Duplicate Elimination in Data Cleaning. *Information Systems*, 26(8):585–606, 2001.

[LLLK99] M. L. Lee, T. W. Ling, H. Lu, and Y. T. Ko. Cleansing Data for Mining and Warehousing. In T. J. M. Bench-Capon, G. Soda, and A. M. Tjoa, editors, *Database and Expert Systems Applications, 10th International Conference, DEXA '99, Florence, Italy, 1999, Proceedings*, volume 1677 of *Lecture Notes in Computer Science*, pages 751–760. Springer, 1999.

[LMR90] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, 1990.

[LSKW02] Y. W. Lee, D. M. Strong, B. K. Kahn, and R. Y. Wang. AIMQ: a Methodology for Information Quality Assessment. *Information and Management*, 40(2):133–146, 2002.

[MBR01] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In P. M. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11–14, 2001, Roma, Italy*, pages 49–58. Morgan Kaufmann, 2001.

[McC87] A.C. McCutcheon. *Latent class analysis.* Sage Publications, Beverly Hills, 1987.

[MHH00] R. J. Miller, L. M. Haas, and M. A. Hernandez. Schema Mapping as Query Discovery. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt*, pages 77–88. Morgan Kaufmann, 2000.

[PS96] G. Piatetsky-Shapiro. Data Mining and Knowledge Discovery in Business Databases. In Z. W. Ras and M. Michalewicz, editors, *Foundations of Intelligent Systems, Proceedings ISMIS '96, LNAI 1079*, pages 56–67. Springer–Verlag, 1996.

[PTU00] L. Palopoli, G. Terracina, and D. Ursino. The System DIKE: Towards the Semi–Automatic Synthesis of Cooperative Information Systems and Data Warehouses. In Y. Masunaga, J. Pokorny, J. Stuller, and B. Thalheim, editors, *Proceedings of Chalenges, 2000 ADBIS–DASFAA Symposium on Advances in Databases and Information Systems, Enlarged Fourth East–European Conference on Advances in Databases and Information Systems, Prague, Czech Republic, September 5–8, 2000*, pages 108–117. Matfyz Press, 2000.

[RB01] E. Rahm and P.A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.

[RD00] E. Rahm and H. H. Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[RH01] V. Raman and J. M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *Proceedings of the 27th International Conference on Very Large Databases (VLDB), September 11–14, Rome,Italy, 2001*, pages 381–390. Morgan Kaufmann, 2001.

[RPG95] M. P. Reddy, B. E. Prasad, and A. Gupta. Formulating Global Integrity Constraints During Derivation of Global Schema. *Data & Knowledge Engineering*, 16(3):241–268, 1995.

[RPRG94] M. P. Reddy, B. E. Prasad, P. G. Reddy, and A. Gupta. A Methodology for Integration of Heterogeneous Databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):920–933, 1994.

[Sch95] I. Schmitt. *Flexible Integration and Derivation of Heterogeneous Schemata in Federated Database Systems.* Preprint Nr. 10, Fakultät für Informatik, Universität Magdeburg, 1995.

[SCS99] G. Saake, S. Conrad, and I. Schmitt. Database Design. *Encyclopedia of Electrical and Electronics Engineering*, 4:540–567, 1999.

[SL90] A.P. Sheth and J.A. Larson. Federated Database Systems for Managing Distributed, Heterogenous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[SN88] M. Schrefl and E.J. Neuhold. Object Class Definition by Generalization Using Upward Inheritance. In *Proceedings 4th International Conference on Data Engineering (ICDE'88)*, pages 4–13. IEEE Computer Society Press, 1988.

[SP94] S. Spaccapietra and C. Parent. View Integration: A Step Forward in Solving Structural Conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258–274, 1994.

[SPD92] S. Spaccapietra, C. Parent, and Y. Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *VLDB Journal*, 1(1):81–126, 1992.

[SS96a] I. Schmitt and G. Saake. Integration of Inheritance Trees as Part of View Generation for Database Federations. In B. Thalheim, editor, *Proceedings of the 15th International Conference on Conceptual Modelling (ER'96)*, number 1157 in Lecture Notes in Computer Science, pages 195–210. Springer–Verlag, 1996.

[SS96b] I. Schmitt and G. Saake. Schema Integration and View Generation by Resolving Intensional and Extensional Overlappings. In K. Yetongnon and S. Hariri, editors, *Parallel and Distributed Computing Systems: Proceedings of the ISCA International Conference (PDCS'96)*, pages 751–758. International Society for Computers and Their Applications, 1996.

[WS96] R. Y. Wang and D. M. Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4):5–34, 1996.

[WS02] K. Winkler and M. Spiliopoulou. Structuring Domain–Specific Text Archives by Deriving a Probabilistic XML DTD. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02), Helsinki, Finland, August 2002*, volume 2431 of *Lecture Notes in Computer Science*, pages 461–474. Springer, 2002.

[YMHF01] L.-L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data–Driven Understanding and Refinement of Schema Mappings. In W. G. Aref, editor, *ACM SIGMOD Conference 2001: Santa Barbara, CA, USA*. SIGMOD 2001 Electronic Proceedings, http://www.acm.org/sigmod/sigmod01/eproceedings, 2001.

[Zmu78] R. Zmud. Concepts, Theories and Techniques: an Empirical Investigation of the Dimensionality of the Concept of Information. *Decision Sciences*, 9(2):187–195, 1978.