Document Number:	DWQ-UNIROMA-002
Project Name:	Foundations of Data Warehouse Quality (DWQ)
Project Number:	EP 22469
Title:	**Source Integration in Data Warehousing**
Authors:	Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati
Workpackage:	WP3
Document Type:	Deliverable
Classification:	Confidential
Distribution:	DWQ Consortium
Status:	Final
Document file:	wp3-1.tex
Version:	1.4
Date:	October 23, 1997
Number of pages:	48

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	September 23, 1997	First draft
v.1.1	October 2, 1997	Restructuring of sections
v.1.2	October 8, 1997	Inclusion of "State of the Art in Source Integration"
v.1.3	October 11, 1997	Section on reasoning completed
v.1.4	October 23, 1997	Final version

1 Introduction

According to [Inm96], integration is the most important aspect of a data warehouse. When data passes from the application-oriented operational environment to the data warehouse, possible inconsistencies and redundancies should be resolved, so that the warehouse is able to provide an integrated and reconciled view of data of the organization.

In DWQ, the ultimate goal of source integration is to represent the migration of the data from the sources to the Data Warehouse, in order to support the design of (materialized) views that meets user requirements, and have high quality with respect to correctness, interpretability, usefulness, believability.

The approach taken in the project is based on building a conceptual representation of both the information sources and the Data Warehouse. An important aspect of the conceptual representation is the explicit specification of the set of interdependencies between objects in the sources and objects in the Data Warehouse. Thus, integration is seen as the process of understanding and representing the relationships between data in the sources and in the Data Warehouse, possibly with some reconciling actions, rather than producing a unified data schema. Moreover, suitable reasoning techniques associated with the conceptual formalism are used to support the designer during the resulting specification process.

The present report illustrates the work carried out in the context of WP3 - Source Integration of DWQ, and is organized as follows. In Section 2, we provide an overview of the state of the art in data integration, in particular within the context of Data Warehousing. In Section 3, we illustrate the main characteristics of our approach to source integration. In Section 4, we describe the formalism we use for expressing the conceptual representation of the Data Warehouse. Section 5 presents the notion of intermodel assertion, and illustrates its use in the source integration framework. In Section 6 we describe the formalism we use for describing the logical content of the sources and the Data Warehouse. In Section 7 we show how the logic-based nature of the formalism allows us to define suitable reasoning techniques supporting source integration. In Section 8 we illustrate the methodological steps for source integration. Section 9 concludes the report, by pointing out the future work within WP3. Finally, the Appendices include relevant papers on the topic addressed in the report.

2 State of the Art in Database Integration

The type of integration to be carried out in the design of a data warehouse can be seen within the field of information integration, which has been investigated in different areas such as:

- databases, where both schema and data integration are a central issue;
- cooperative information systems, where integration is one of the problems arising when component information systems cooperate in order to perform their tasks, for example by sharing data;
- global information systems, where heterogeneous information sources are put online, e.g. on the Web, and accessed in an integrated way;
- knowledge representation, where typically integration is considered in the more general context of conforming and merging logical theories or other kinds of knowledge expressed in a more general form than just data.

Here, we deal with the first area, which is the appropriate one to look at in the context of data warehouses. A large body of work has been carried out in recent years in data integration [Hul97, Ull97]. A first classification of the various approaches regards the context of integration, and, consequently, the kind of inputs and outputs of the integration process, and the goal of the process itself. We distinguish among schema integration, virtual data integration, and materialized data integration.

1. Schema integration. In this case the input of integration is a set of data schemas (source schemas), and the output is one data schema (target schema) representing the reconciled intensional representation of all input schemas. The output includes also the specification of how to map source data schemas into portions of the target schema.

2. Virtual data integration. The input is a set of source data sets, and the output is a specification of how to provide a global and unified access to the sources in order to satisfy certain information needs, without interfering with the autonomy of the sources.

3. Materialized data integration. As in the previous case, the input is a set of source data sets, but here the output is a data set representing a reconciled view of the input sources, both at the intensional and the extensional level.

The first context is meaningful, for example, in database design, and in general in all the situations where descriptions of the intensional level of information sources are to be compared and unified. The second one has been adopted in several multidatabase and distributed database projects. The third one is meaningful, for example, both in information system re-engineering and data warehousing.

Although the third context is the appropriate one for investigating the integration process to be carried out in the design of a data warehouse, work on both schema and virtual data integration provides several interesting concepts and notions that are useful for the third context too. This is the reason why we include such research in our analysis.

The rest of this section presents an overview on the state of the art in data integration in the context of Data Warehousing, both in research and in practice.

2.1 State of the Art – Research

We concentrate on the work on schema and data integration that is most relevant for data warehousing and distinguish contributions treating:

1. only schema integration,

2. data integration with virtual views,

3. data integration with materialized views.

Schema integration must either implicitly or explicitly be done in order to integrate data. This makes work on schema integration relevant to any data integration approach and in particular to the context of data warehousing. In reviewing relevant work in the context of schema integration (Section 2.2), we make use of several key aspects, which can be summarized as follows:

A1 Whether a global schema is produced or not.

A2 Which is the schema integration methodological step the work refers to (preintegration, schema comparison, schema conforming, schema merging and restructuring).

A3 Which is the formalism for representing data schemas.

A4 To what extent the notion of data schema quality is taken into account (correctness, completeness, minimality, understandability [BLN86]).

Data integration with virtual views and with materialized views are sometimes considered as complementary approaches. In a virtual views approach data are kept only in the sources and are queried using the views. Hence much emphasis is put on query decomposition, shipping, and reconstruction: A query needs to be decomposed in parts, each of which is shipped to the corresponding source. The data returned are then integrated to reconstruct the answer to the query.

When discussing papers dealing with virtual data integration (Section 2.3), we consider the following additional points:

B1 Whether a global view is assumed or not.

B2 Which is the virtual integration methodological step the work refers to (query decomposition, shipping, reconstruction).

B3 Which is the formalism for representing data (file, legacy, relational DB, OODB, unstructured).

B4 Which query language is used for posing global queries.

B5 Which data matching criteria are adopted (key-based, look-up table-based, comparison-based, historical-based).

B6 To what extent the notion of data quality is taken into account (interpretability, usefulness, accessibility, credibility – see chapter on quality).

In a materialized views approach, views are populated with data extracted from the sources. Query decomposition, shipping and reconstruction are used in the phase of populating views rather than in the process of query answering. Additionally, issues like maintenance strategies and timings, that are absent in the context of virtual views, become crucial for materialized views. We remark that most of the issues that arise in the approaches based on virtual views are of importance for materialized views approaches as well. So, despite differences in focus, we take the position that materialized views are conceptually a "specialization" of virtual views.

When reviewing papers dealing with materialized data integration (Section 2.4), the following further aspects are taken into account (partly borrowed from [HZ96]):

C1 Which data are materialized.

C2 Activeness of the sources (sufficient, restricted, non active).

C3 Maintenance strategy (local incremental update, polling-based, complete refresh) and timing.

C4 To what extent the notion of data quality is taken into account (interpretability, usefulness, accessibility, credibility – see chapter on quality).

2.2 Schema Integration

Schema Integration is the activity of integrating the schemas of the various sources in order to produce a homogeneous description of the data of interest. This activity is traditionally performed in one-shot, resulting in a global schema in which all data are represented uniformly [BLN86]. More recently, in order to deal with autonomous and dynamic information sources, an incremental approach is arising [CL93]. Such an approach consists of building a collection of independent partial schemas, formalizing the relationships among entities in the partial schemas by means of the so-called *interschema assertions*. In principle, under the assumption that the various information sources remain unchanged, the incremental approach would eventually result in a global schema, similar to those obtained through a traditional one-shot approach, although in practice, due to the dynamics of the sources, such a result is never achieved. Additionally, the integration may be partial, taking into account only certain aspects or components of the sources [CL93].

Independently from the integration strategy adopted, schema integration is divided in several methodological steps, with the aim of relating the different components of schemas, finding and resolving conflicts in the representation of the same data among the different schemas, and eventually merging the conformed schemas into a global one. In [BLN86], the following methodological steps are singled out:

1. preintegration
2. schema comparison
3. schema conforming
4. schema merging and restructuring.

We review recent studies on schema integration, according to the steps they address. We refer to [BLN86] for a comprehensive survey on previous work in this area.

2.2.1 Preintegration

Preintegration consists of an analysis of the schemas to decide the general integration policy: choosing the schemas to be integrated, deciding the order of integration, and possibly assigning preferences to entire schemas or portions thereof. The choices made in this phase influence the usefulness and relevance of the data corresponding to the global schema. During this phase also additional information relevant to integration is collected, such as assertions or constraints among views in a schema. Such process is sometimes referred to as *semantic enrichment* [GSSC95b, GSSC95a, BIG94, RPRG94]. It is usually performed by translating the source schemas into a richer data model, that allows for representing information about dependencies, null values, and other semantic properties, thus increasing interpretability and believability of the source data.

For example, [BIG94] enriches relational schemas using a class based logical formalism, a Description Logic, available in the terminological system BACK [Pel91]. Instead, [GSSC95b, GSSC95a] use as unifying model a specific object-oriented model with different types of specialization and aggregation constructs.

[SGN93] proposes the creation of a knowledge base (terminology) in the preintegration step. More precisely, a hierarchy of attributes is generated, thus representing the relationship among attributes in different schemas. Then, source schemas are classified: the terminology thus obtained corresponds to a partially integrated schema. Such a terminology is then restructured by using typical reasoning services of class-based logical formalisms. The underlying data model is hence the formalism used for expressing the terminology: more precisely, a description logic (CANDIDE) is used.

[Joh94] defines a collection of transformations on schemas represented in a first order language augmented with rules to express constraints. Such transformations are correct with respect to a given notion of information preservation, and constitute the core of a "standardization" step in a new schema integration methodology. This step is performed before schema comparison and logically subsumes the schema conforming phase, which is not necessary in the new methodology.

2.2.2 Schema Comparison

Schema comparison consists of an analysis to determine the correlations among concepts of different schemas and to detect possible conflicts. Interschema properties are typically discovered during this phase.

The types of conflicts that arise when comparing source schema components have been studied extensively in the literature (see e.g. [BLN86, KLK91, SPD92, ON94, RPRG94]) and consensus has arisen on their classification, which can be summarized as follows:

- *Heterogeneity conflicts* arise when different data models are used for the source schemas.
- *Naming conflicts* arise because different schemas may refer to the same data using different terminologies. Typically one distinguishes between *homonyms*, where the same name is used to denote two different concepts, and *synonyms*, where the same concept is denoted by different names.
- *Semantic conflicts* arise due to different choices in the level of abstraction when modeling similar real-world entities.
- *Structural conflicts* arise due to different choices of constructs for representing the same concepts.

In general, this phase requires a strong knowledge of the semantics underlying the concepts represented by the schemas. The more the semantics is represented formally in the schema, the easier can similar concepts in different schemas be automatically detected, possibly with the help of specific CASE tools that support the designer. Traditionally, schema comparison was performed manually [BLN86]. However recent methodologies and techniques emphasize automatic support to this phase.

For example, [GLN92] proposes an architecture where schema comparison and the subsequent phase of schema conforming are iterated. At each cycle the system proposes correspondences between concepts that can be confirmed or rejected by the designer. Newly established correspondences are used by the system to conform the schemas and to guide its proposals in the following cycle. Both the component schema and the resulting global schema are expressed in a data model that essentially corresponds to an Entity-Relationship model extended with complex objects.

[BIG94] exploits the reasoning capabilities of the terminological system to classify relational schema components and derive candidate correspondences between them expressed in the description logic BACK.

[BN97] introduces new types of reasoning capabilities for Description Logics, called unification and matching of concepts, which might support the designer in detecting naming conflicts and semantic conflicts, provided that (a part of) the schema is expressed in Description Logics.

[MYR94] studies the problem of deciding equivalence and dominance between schemas, based on a formal notion of information capacity given in [Hul86]. The schemas are expressed in a graph-based data model which allows for the representation of inheritance and simple forms of integrity constraints. First, it is shown that such a problem is undecidable in schemas that occur in practice. Then *sufficient* conditions for schema dominance are defined, which are based on a set of schema transformations which preserve schema dominance. A schema S_1 is dominated by a schema S_2 if there is a sequence of such transformations that converts S_1 to S_2.

Reconciliation of semantic discrepancies in the relational context due to information represented as data in one databases and as meta-data in another, is discussed in [KLK91]. The paper proposes a solution based on reifying relations and databases by transforming them into a structured representation.

In [BCW90] schema comparison in an extended Entity-Relationship model is performed by analyzing structural analogies between subschemas through the use of similarity vectors. Subsequent conforming is achieved by transforming the structures into a canonical form.

2.2.3 Schema Conforming

The goal of this activity is to conform or align schemas to make them compatible for integration. The most challenging aspect is represented by conflict resolution which in general cannot be fully automatized. Typically, semi-automatic solutions to schema conforming are proposed, in which human intervention by the designer is requested by the system when conflicts have to be resolved. Recent methodologies and techniques emphasize also the automatic resolution of specific types of conflicts (e.g. structural conflicts). However, a logical reconstruction of conflict resolution is far from being accomplished and still an active topic of research.

[Qia96] studies the problem of establishing correctness of schema transformations on a formal basis. More specifically, schemas are modeled as abstract data types, and schema transformations are expressed in terms of signature interpretations. The notion of schema transformation correctness is based on a refinement of Hull's notion of information capacity [Hul86]. In particular, such a refinement allows for a formal study of schema transformations between schemas expressed in different data models.

[VW94] presents a general methodology for schema integration, in which the semantics of updates is preserved during the integration process. More precisely, three steps are defined: combination, restructuring, and optimization. In the combination phase, a combined schema is generated, which contains all source schemas and assertions (constraints) expressing relationship among entities in different schemas. The restructuring step is devoted to normalizing (through schema transformations) and merging views, thus obtaining a global schema, which is refined in the optimization phase. Such a methodology is based on a semantic data model which allows the declaration of constraints containing indications on what to do when an update violates that constraint. A set of schema transformations are defined which are *update semantics* preserving, in the sense that any update specified against the transformed schema has the same effect as if it was specified against the original schema.

2.2.4 Schema Merging and Restructuring

During this activity the conformed schemas are superimposed, thus obtaining a (possibly partial) global schema. Such a schema is then tested against qualities such as completeness, correctness, minimality and understandability. This analysis may give rise to transformations of the schema obtained.

[BDK92] defines a technique for schema merging, which consists of a binary merging operator for schemas expressed in a general data model. Such an operator is both commutative and associative, hence the resulting global schema is independent of the order in which the merges are performed.

[SPD92] presents a methodology for schema integration which allows for automatic resolution of structural conflicts and building of the integrated schema without requiring conforming of the initial schemas.

The methodology is applicable to various source data models (relational, Entity-Relationship, and object-oriented models), and is based on an expressive language to state interschema assertions that may involve constructs of schemas expressed in different models. Data model independent integration rules that correspond to the interschema assertions are defined in the general case and are also specialized to the various classical data models. Quality issues are addressed in an informal way. In particular, correctness is achieved by selecting, in case of conflicts, the constructs with weaker constraints. The methodology includes strategies that avoid to introduce redundant constructs in the generated schema. Completeness, however is not guaranteed since the model adopted for the global schema lacks a generalization construct.

[GPNS92, GPC+92] present an integration technique (*structural integration*) which allows for the integration of entities that have structural similarities, even if they differ semantically. An object-oriented model, called DUAL model, is used, in which structural aspects are represented as object types, and semantic aspects are represented as classes. Two notions of correspondence between classes are defined: full structural correspondence and partial structural correspondence. The (partial) integration of two schemas is then obtained through a generalization of the classes representing the original schemas.

2.3 Data Integration – Virtual

While precise methodologies have been developed for schema integration, there is no consensus on the methodological steps for data integration, with either virtual or materialized views. We use as guiding criteria to survey the state of the art in this area the various approaches and implementation efforts in data integration. For virtual views we focus on those aspects that are most relevant to data warehousing.

One of the aspects distinguishing data integration from schema integration is that of *object matching*, i.e. establishing when different objects in different source databases represent in fact the same real world element and should therefore be identified. Such an aspect has a strong impact on data quality dimensions like redundancy and accuracy. The most simple object matching criterion, called key-based, is to identify objects having the same key. Lookup-table and identity function based approaches use more complex criteria to match objects.

[Alb96] argues that object identity should be an equivalence relation, and therefore proposes to represent objects in the integrated schema as equivalence classes. If one wants to respect this requirement, the evaluation of queries against the integrated schema may however become more complex. In fact, in the presence of subtype relationships, even simple non-recursive queries against the integrated schema may have to be translated to recursive queries against the underlying data, reflecting the transitive property of the equivalence relation.

2.3.1 Carnot

In the *Carnot* system described in [CHS91, HJK+93] individual schemas are mapped onto a large ontology which is provided by the *Cyc* knowledge base [LG90]. Such an ontology is expressed in an extended first order representation language called *Global Context Language* (GCL). The high expressiveness of GCL allows to represent in the global Cyc schema both meta-models for various schema formalisms and all available knowledge about the individual schemas, including integrity constraints, allowed operations, and organizational knowledge. The mapping between each model and the global context involves both syntax and semantics and is stated in terms of so called *articulation axioms* which play the role of interschema assertions. Once the mapping is established, queries are answered and updates are performed by first translating them to GCL and then distributing them to the different resource management systems.

A similar architecture also based on a first order formalism for the unified schema is proposed in [DW91]. Source schemas and global views are represented in a knowledge base, and an inference engine based on rules is used to extract integrated information from the sources.

2.3.2 SIMS

[ACHK93, AKC96] present SIMS, a system for data integration from multiple information sources. Instead of performing schema merging of the sources, SIMS adopts an alternative approach, in which first a *domain model* of the application domain is defined, and then each source is described using this model. Notably, in SIMS there is no fixed mapping from a query to the sources: sources are dynamically selected and integrated when the query is submitted. This allows for handling dynamic information sources, reacting to newly available pieces of information and unexpectedly missing ones. The ability to integrate information sources that are not databases is also pointed out.

The domain model is formalized in terms of a class-based representation language (LOOM). Query processing is performed by using four basic components: (1) a query reformulation component, which identifies the sources required in order to answer the query and the data integration needed; (2) a query access planning component, which builds a plan for retrieving the information requested by the reformulated query; (3) a semantic query-plan optimization component, which both learns the rules for optimizing queries and uses semantic optimization techniques to support multidatabase queries; (4) an execution component, which executes the optimized query plan.

2.3.3 Information Manifold

The *Information Manifold* (IM) is an implemented system, developed at AT&T, for information gathering from disparate sources such as databases, SGML documents, unstructured files [LSK95, KLSS95, LRO96]. It is based on a rich domain model, expressed in a knowledge base, that allows for describing various properties of the information sources, from the topics they are about to the physical characteristics they have. This enables the user to pose high-level queries to extract the information from different sources in a unified way. Observe that the ability of modeling both conceptual and physical characteristics increases believability and accessibility of data.

In IM two components are identified: a *world view*, and an *information source description* for each source. Both the world view and the information source descriptions are formed essentially by relational schemas. Observe that, although the information sources can be of any kind and not necessary relational database, a view of their data in terms of relations needs to be provided.

The relational schemas of the world view and the information sources are enhanced by sophisticated type descriptions of the attributes of the relations, formulated using the simple description logic of the CLASSIC system [PMB$^+$91]. This allows for forming natural hierarchies among types, reflecting both semantic and structural information. Moreover various forms of automatic reasoning for query optimization are possible using the inference procedures of description logics. Constraints involving relational schemas and type descriptions of both the world view and the information source descriptions are expressed as Datalog rules in which both relations and type descriptions (interpreted as unary relations) may occur.

In IM queries are formulated against the world view and answers involve retrieving information from several sources. IM uses as query language Datalog enhanced with type descriptions in CLASSIC. Using automatic reasoning on such type descriptions, IM supports optimization techniques for query decomposition that aim at the minimization of the information sources involved in answering a global query, by isolating for each sub-query those sources that are *relevant*.

It is also possible to specify that an information source has *complete* information on the domain it represents.

The architecture of IM suits the dynamic nature of the information sources. In particular, to add a new source, only its description (i.e. its view as a relational schema and the related types and constraints) needs to be added, without changing the existing integration mechanisms of the system.

It should be stressed that IM does not allow any form of update in either the information sources or the world view. Updates on the information sources are external to IM, while propagation of updates from the world view to the single sources is not supported.

Finally, although IM does not implement any specific schema integration strategy, it implicitly enforces one based on interschema assertions, with a partial global schema. Indeed IM supports a collection of schemas, one for each source, plus one for the world view. The source schemas are related to the world view by means of constraints which can in fact be seen as interschema assertions. This strategy allows for an incremental integration of the information sources and it is well suited to deal with the their dynamic nature.

2.3.4 TSIMMIS

TSIMMIS (*The Stanford-IBM Manager of Multiple Information Sources*) is a project that shares with IM the goal of providing tools for the integrated access to multiple and diverse information sources and repositories [CGMH+94, Ull97].

Each information source is equipped with a *wrapper* that encapsulates the source, converting the underlying data objects to a common data model. In TSIMMIS a simple object oriented data model called *Object Exchange Model* (OEM) is used. On top of wrappers TSIMMIS has another kind of system components: the *mediators* [Wie92]. Each mediator obtains information from one or more wrappers or other mediators, refines this information by integrating and resolving conflicts among the pieces of information from the different sources, and provides the resulting information to the user or other mediators in turn. Mediators can be conceptually seen as views of data found in one or more sources which are suitably integrated and processed. The mediator is defined in terms of a logical language called MSL, which is essentially Datalog extended to support OEM objects. Typically mediators realize virtual views since they do not store data locally. However it is possible that some mediator materializes the view it provides.

When a user poses a query to the system, a specific mediator is selected. Such a mediator decomposes the query and propagates the resulting sub-queries to the levels below it (either wrappers or mediators). The answer provided by such levels is then reconstructed performing integration steps and processing (using out-of-the-shelf techniques). The TSIMMIS query language is an SQL-like language adapted in order to treat suitably OEM objects.

Adding a new information source to TSIMMIS requires to build a wrapper for the source and to change all mediators that will use the new source. The research within the TSIMMIS project has devised techniques for automatically generating both wrappers and mediators.

It has to be stressed that no global integration is ever performed. Each mediator performs integration independently. As a result, for example, a certain concept may be seen in completely different and even inconsistent ways by different mediators. Such form of integration can be called *query-based*, since each mediator supports a certain set of queries, i.e. those related to the view it provides.

2.4 Data Integration – Materialized

The context of data integration with materialized views is the one most closely related to data warehousing. Maintenance of views against updates to the sources is a central aspect in this context and the effectiveness of maintenance affects timeliness and availability of data. In particular, recomputing the view entirely from scratch is often expensive and makes frequent refreshing of views impractical. The study of conditions for reducing overhead in view recomputation is an active research topic. E.g. [GJM96] introduces the notion of self-maintainability: Self-maintainable views are materialized views which can be updated directly using only log files of the sources.

However, those issues related to the choice of views to materialize and to the maintenance of materialized views are outside the scope of this chapter, since they pertain more to the design phase of the data warehouse. Here, as in the previous case, we concentrate on aspects related to data integration, and proceed by reviewing the current approaches and implementation efforts.

2.4.1 The Squirrel Project

The *Squirrel* Project is a system under development at the University of Colorado [ZHKF95b, ZHKF95a, ZHK96, HZ96] which provides a framework for data integration based on the notion of *integration mediator*. Integration mediators are active modules that support data integration using a hybrid of virtual and materialized data approaches.

In [ZHKF95a, ZHK96], emphasis is posed on data materialization, and Squirrel mediators are defined, which consist of software components implementing materialized integrated views over multiple sources. A key feature of such components is their ability to incrementally maintain the integrated views by relying on the active capabilities of sources. More precisely, at startup the mediator sends to the source databases a specification of the incremental update information needed to maintain the views and expects sources to actively provide such information.

The architecture of a Squirrel mediator consists of three components: a set of active rules, an execution model for such rules, and a View Decomposition Plan (VDP). The notion of VDP is analogous to that of Query Decomposition Plan in query optimization. More specifically, the VDP specifies the classes that the mediator maintains, and provides the basic structure for supporting incremental maintenance.

Moreover, an automatic generator of Squirrel integrators has been developed. Such a module takes as input a specification of the mediator expressed in a high-level Integration Specification Language (ISL). A specification in ISL includes a description of the relevant subschemas of the source databases, and the match criteria between objects of families of corresponding classes, in particular a list of the classes that are matched and a binary matching predicate specifying correspondences between objects of two classes. The output of this module is an implementation of the mediator in the *Heraclitus* language, a database programming language whose main feature is the ability of representing and manipulating collections of updates to the current database state (deltas). In [ZHKF95b] the language *H20* is used, which is an object-oriented extension of Heraclitus.

[ZHKF95b] addresses the problem of object matching in Squirrel mediators. In particular, a framework is presented for supporting intricate object identifier (OID) match criteria, among which key-based matching, lookup-table-based matching, historical-based-matching, and comparison-based matching. The last criterion allows for both considering attributes other than keys in object matching, and using arbitrary boolean functions in the specification of object matching.

An extension of the Squirrel framework in which both partially materialized and fully virtual integrated views are supported is presented in [HZ96]. Emphasis is posed on the fact that, in several applications, materializing only matching information is a suitable design choice. Moreover, data quality is considered by defining formal properties of consistency and *freshness* for integrated views, and showing that automatically generated Squirrel mediators satisfy such properties, when the sources and the network respect some "natural" assumptions.

Finally, a taxonomy of the solution space for data integration is presented, emphasizing situations where part or all of the information integrated is materialized.

2.4.2 WHIPS

The WHIPS (*WareHouse Information Prototype at Stanford*) system [HGMW+95, WGL+96] is a data warehouse system currently under development at Stanford University which is being used as a test-bed for various integration schemes. The WHIPS architecture consists of a set of independent modules implemented as CORBA objects. This ensures both modularity and scalability. The central component of the system is the *integrator*, to which all other modules report. Different data models may be used both for each source and the data warehouse data. The relational model is used as a unifying model, and for each source and the data warehouse the underlying data are converted to the relational model by a specific *wrapper*.

Views are defined by the system administrator in a subset of SQL that includes select-project-join views. The view definition is passed to the *view-specifier* module which parses it into an internal structure, called *view-tree* that includes also information from a *metadata store*. The view-tree is sent to the integrator which spawns a *view manager* that is responsible for managing the view at the data warehouse. The view manager initializes the view upon notification by the integrator and computes the changes to the view that become necessary due to updates of the sources. For each source such updates are detected by a *monitor* which forwards them to the integrator who notifies the relevant view managers. The update by the view manager is done by passing appropriate queries to a global *query processor*. The answers that are sent back are adjusted and combined according to view consistency and maintenance algorithms [ZGMHW95], and are then forwarded to the warehouse wrapper.

As already mentioned, the central component of the system is the integrator, whose main role is to facilitate view maintenance. The integrator uses a set of rules that are automatically generated from the view tree to decide which source updates are relevant for which views and therefore have to be forwarded to the corresponding view managers. The current implementation uses a naive strategy which dictates that all modifications to a relation over which a view is defined are relevant to the view. However, this does not take into account e.g. selection conditions, which may render an update irrelevant to a view. Extensions of the integrator in such directions are under development.

2.5 State of the Art – Practice

The market offers a large variety of methodologies, tools and products under the label "Data Warehouse". The first distinction to be made is between:

- data warehouse engines,
- tools supporting specific tasks of the data warehouse management,
- methodologies for data warehouse construction.

Under the category of data warehouse *engines* one finds the basic DBMS hardware and software technology adapted to the needs of data warehouse applications, which essentially amounts to increased performance in terms of size of the database, loading and query response times.

With regard to Data Warehouse tools, it is not easy to isolate those aspects that are specific to the tasks of source integration and data reconciliation, that are the main focus of this chapter of the report. Consequently, we provide a general introduction, highlighting those aspects that are related to the two above mentioned tasks. In particular, we deal with the main classes of tools for data warehousing that one can find in the marketplace. It is worth noticing that under the category of data warehouse *solutions* there are several collections of tools that can be provided by the same vendor, but can also be put together including third parties' components which are available as *separate* products.

The tools can be roughly divided in tools for data analysis, data extraction and cleaning, warehouse manager. After a discussion on the above tools, we briefly address the issue of methodologies for data warehouse construction, that are supported along with data warehouse solutions.

2.5.1 Tools for Data Analysis

The main functionalities of this category of tools are: multidimensional modeling, graphical interface, different presentation of data, statistical analysis functions, report generation. One functionality that sometimes comes as a separate tool is that of data mining, which enables for the discovery of unknown relationships among data. *Executive Information Systems* follow in this category and in fact this technology has been leading during the first developments of data warehousing. The tools for data analysis can be oriented to the development of small size *data marts* (data aggregated for a specific data analysis task often not to be used after the accomplishment of the task) or else support larger data warehouse applications.

A basic assumption underlying the tools for data analysis is that data are available from a data warehouse, typically in the form of a relational database and an application specific multidimensional model is built supporting different forms of data aggregation. Nowadays, it has been recognized that creation and maintenance of large data warehouses is of primary importance for exploiting the possibilities and managing the complexities involved in data analysis. Recent figures show that the most relevant portion of investments in new data warehouse products will be in this direction.

2.5.2 Data Extraction and Cleaning

This category of tools is specifically devoted to the interface with the operational databases. Extraction tools may be further divided in software development environments, whose capabilities range from automatic software generation to more traditional programming environments, to query handlers which can deal with a variety of query languages, typically SQL dialects.

In the first case, the user is required to provide, typically through an interactive session, the mapping between the data to be extracted from the information sources and the form they take in the data warehouse. Otherwise, the procedures for extracting the data from the sources need to be hand coded. It is worth noting that if the mapping from the sources to the target is not a one-to-one mapping the possibility of adding hand coded extraction procedures is left as the only possible way out.

Several forms of data transformation and data cleaning can be associated with data extraction:

- *data integration*: integration of different data formats, codes and values, based on predefined mappings;
- *data validation* identification of potentially inconsistent data that can be either removed or fixed;
- *data filtering* filtering of the data according to the requirements of the data warehouse.

In addition, data mining is sometimes suggested at this stage, thus avoiding the transfer of large quantities of data to the tools for data analysis.

The products for data transformation can either filter the data while they are loaded in the data warehouse or perform a database auditing function, after the data have been loaded. Often, several kinds of transformation can be supported by a single tool. In general, the tools for data transformation and cleaning are quite successful, although they are rather limited in their scope, since they are restricted to specific kinds of data, i.e. postal codes, addresses etc..

Here is a list of tools that fall into this category: Carleton PassPort of Carleton Corporation, EXTRACT of Evolutionary Tech., Metacube Agents of Informix, Sybase Mpp, Datajoiner of IBM. In addition, The Warehouse Manager of Prism Solutions, and SAS Warehouse Manager of SAS Institute include internal facilities for these tasks. Finally, among the tools for data mining we have Integrity of Vality Tech. and Wizrule of Wizsoft.

2.5.3 Warehouse Manager

The Warehouse Manager is normally present in every data warehouse solution. However there is a great variety of proposals that differ with respect to the role played by the Manager. Clearly, there is the need to provide a common basis for both the data warehouse construction process and the access and analysis of data. However, there is a stronger attention towards the first aspect, since most of the tools for data access and analysis rely on a standard database system as the enterprise data warehouse (as long as it provides enough capacity of data storage and handling). The main function of the Warehouse Manager is to handle the metadata (i.e. data schemas). Consequently, a large portion of the proposals for data warehouse management arise from the companies that have been supporting tools and methodologies for data modeling. Among the metadata we find a description of the information sources, of the data warehouse itself and of the mapping between the two.

In addition, the Warehouse Manager is responsible for handling the extraction, update and maintenance process. The task of data extraction has been separately addressed since it is often accomplished by specialized tools. A Warehouse Manager allows the user to specify a scheduling for the extraction of data and update policies. This functionality can either be automatically enforced from a high level user specification, typically based on a set of predefined choices, or must be explicitly coded by the user.

Another functionality that is sometimes supported by the Warehouse Manager is the connection to the tools for data analysis. In this respect the Warehouse Manager can handle metadata about the data organization used by the tools for data analysis. Clearly, this is important to keep a connection between the models used for data analysis and the models of the data in the data warehouse and, consequently, in the information sources. In particular, this makes it possible to provide a unified framework for handling the design and transformation of the various data models that are needed in data warehousing.

Among the tools that support data warehouse management we can identify a first group which is rooted on conventional data modeling and provides tools for reverse engineering and schema integration. Among them we find Groundworks of Cayenne (Bachman), ERWIN ERX of LogicWorks, and Kismetic Analyst of Kismeta. A second group of solutions is more focused on the construction and maintenance of the data warehouse, among them The Warehouse Manager of Prism Solutions, SAS Warehouse Manager of SAS Institute, the Metacube Warehouse Manager of Informix, and Sourcepoint of Software AG.

2.5.4 Methodologies for Data Warehouse Construction

As already noticed, the problem of data warehouse construction is being specifically addressed by the tools for data warehouse management. Being the data warehouse construction a special case of database construction, the methodologies and tools that are proposed in this context have their roots in the tradition of database design. The most significant feature required for data warehouse construction is the ability to deal with several information sources. Two main approaches have been proposed:

1. construction of an integrated enterprise schema (usually called enterprise model);
2. explicit mapping of the relationship between the data warehouse and the sources.

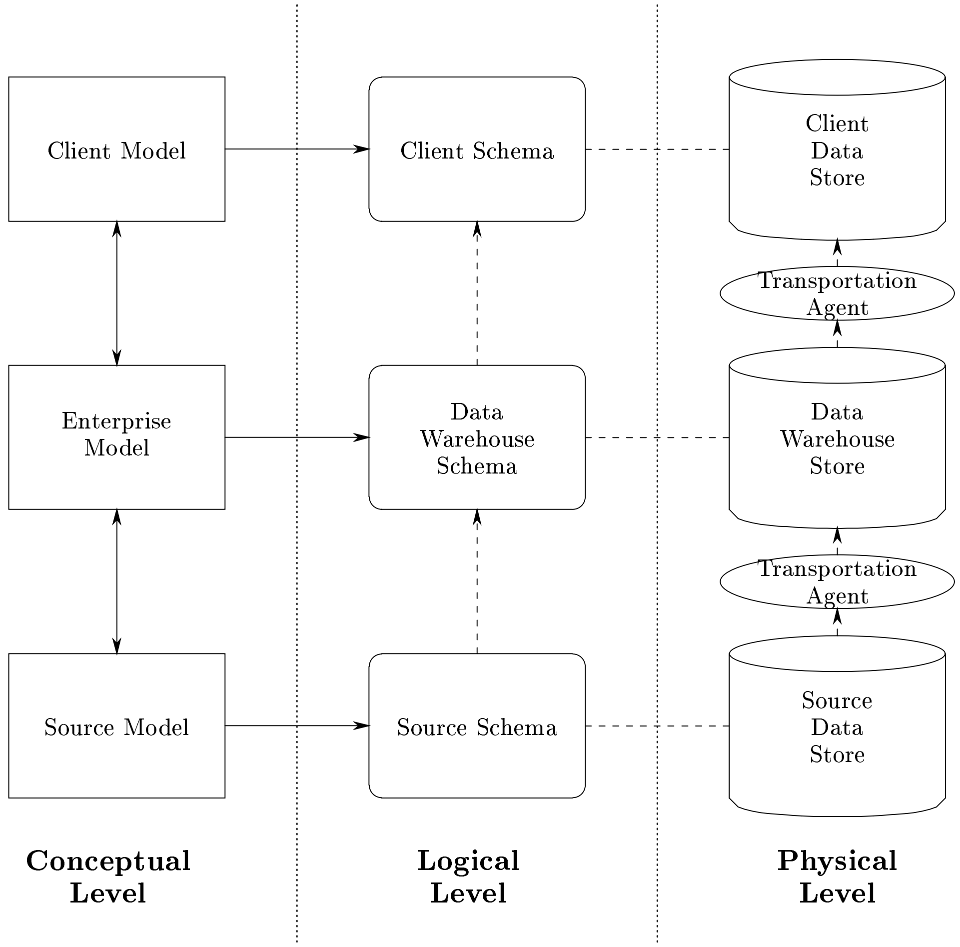

Figure 1: The DWQ Architecture

The first approach is proposed by those vendors that have built tools and associated methodologies for enterprise modeling. Without entering in a detailed analysis of the methodologies of specific tools, we remark that the most extensively supported model is the Entity-Relationship model, which can sometimes be derived through reverse engineering tools. In addition, some of the tools can deal with the star and snowflake models, but there is little support to model transformation from standard models to data warehouse specific ones. For a more detailed description of these aspects see the chapter on architecture.

With respect to the second approach, several kinds of relationships can be established between different schemas (we call them *interschema assertions*), in particular between the schemas of the enterprise and those of the individual sources. For example, the notion of *proprietary* entity has been used to specify that the complete information about the extension of a certain entity is found within a certain information source. On the other hand, graphical and structure directed editors guide the user in the specification of the relationship between the data warehouse and the information sources. However, there seems to be no support to the verification of the validity of interschema assertions, and, more generally, of the specified relationships.

3 Source Integration in DWQ

The goal of this section is to describe the basic characteristics of our approach to source integration. The next sections will address in further details the various aspects pointed out here.

Generally speaking, if we refer to the DWQ framework illustrated in the Deliverable 2.1, and reported

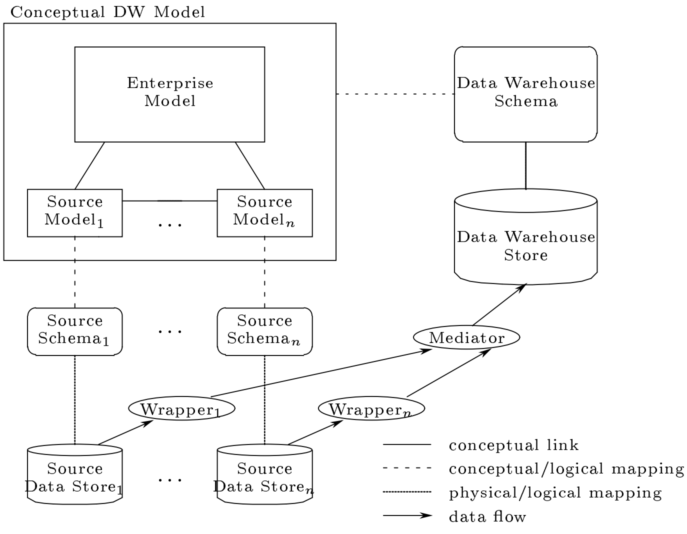

Figure 2: Architecture for Data Integration

here in Figure 1, source integration deals with the objects named "Enterprise Model", "Source Model", "Data Warehouse Schema", and "Source Schema", and is characterized by the following elements (see Figure 2):

- The *Enterprise Model* is a conceptual representation of the global concepts and relationships that are of interest to the Data Warehouse application.
- For a given information source S, the *Source Model* of S is a conceptual representation of the data residing in S.
- The logical content of each source S, called the *Source Schema*, is provided in terms of a set of definitions of relations, each one expressed in terms of a query over the Source Model of S.
- The logical content of the materialized views constituting the Data Warehouse, called the *Data Warehouse Schema*, is provided in terms of a set of definitions of relations, each one expressed in terms of a query over the *Conceptual Data Warehouse Model*.
- Each element of the integration process has a corresponding representation in the DWQ Repository (see Deliverable 2.1).

The current work focuses on the conceptual and logical representation of relations and concepts without concern to the notion of aggregation. Indeed, aggregated data are of a special interest in the Client Model of the DWQ architecture, and the problem of expressing them at the conceptual level will be the subject of future work within WP3.

As we will see in the sequel, the proposed approach to integration is incremental: no "Complete Enterprise Model"[1] is prescribed. Instead, whenever a new portion of a source is taken into account, the new information is integrated with the Enterprise Model, and the necessary new relationships are added. Thus, the Enterprise Model provides a consolidated view of the concepts and the relationships that are important to the enterprise, and have been currently analyzed. Such a view is subject to changes and additions as the analysis of the information sources proceeds. This approach would also allow the notion of Complete Enterprise Model (in particular in the extreme case where all aspects of all sources have been considered), but is more flexible.

As we will see in Section 8, the methodology and the reasoning techniques for assisting source integration support the incremental building of the conceptual and the logical representation of the Data Warehouse, and provide several information to the designer, including:

[1]The notion of Complete Enterprise Model corresponds to the one of global schema used in Database integration.

- which global concepts are relevant for new information requirements
- from which sources can a new view be defined
- which correspondences exist between sources/views
- trace of the integration steps.

The methodology builds on the so called *Integration Knowledge Base* (IKB) that stores information about source integration, and forms a part of the architecture model encoded in the DWQ Repository.

Many of the research results discussed in the previous section can be used in our context, such as results on conflict classification, conflict detection, conflict resolution, schema merging, wrapper design, object matching, etc. On the other hand, the analysis that we have carried out shows that a general and unified support for incremental Data Warehouse integration with concern on data quality is missing.

In the rest of this section, we delve into more details about the basic elements we just described.

3.1 Source Models and Enterprise Model: the Conceptual Level

As we said before, the Enterprise Model is a conceptual representation of the global concepts and relationships that are of interest to the Data Warehouse application. The Enterprise Model corresponds roughly to the notion of integrated conceptual schema in the traditional approaches to schema integration. However, in our context, the Enterprise Model is not necessarily a complete representation of all the data of the sources. Being the integration process incremental, the Enterprise Model provides a consolidated view of the concepts and the relationships that are important to the enterprise, and have been currently analyzed. Such a view is subject to changes and additions as the analysis of the information sources proceeds.

We use a logic-based formalism for describing the Enterprise Model. The formalism is general and powerful enough to express the usual database models, such as the Entity-Relationship Model, the Relational Model, the Object-Oriented Model. Moreover, there are suitable inference techniques associated with the formalism, which allow for carrying out several reasoning services on the representation.

For a given information source S, the Source Model of S is a conceptual representation of the data residing in S, or at least of the portion of data currently taken into account. Again, our approach does not require a source to be fully analyzed and conceptualized. Source Models are expressed by means of the same formalism used for the Enterprise Model.

The formalism we use for expressing both the Enterprise Model and the Source Models is presented in Section 4. The associated reasoning techniques are described in Section 7.

The Enterprise Model, the various Source Models, and the interdependencies between elements of different Source Models and between Source Models and the Enterprise Model, constitute what we call the *Conceptual Data Warehouse Model*. The notion of interdependencies is a central one in our approach. Since the sources are of interest in the overall architecture, integration does not simply mean producing the Enterprise Model, but rather to be able to establish the correct relationships both between the Source Models and the Enterprise Model, and between the various Source Models. We formalize the notion of interdependency by means of interschema assertions (here called intermodel assertions), introduced in [CL93]. An *intermodel assertion* states that one object (i.e. class, entity, or relation) belonging to a certain Model (either the Enterprise or a Source Model) is always a subset of an object belonging to another Model. This simple declarative mechanisms has been showed to be extremely effective in establishing relationships among different database schemas (see also [Hul97]). We use again a logic-based formalism to express intermodel assertions, and the associated inference techniques provide a means to reason about interdependencies among models. Intermodel assertions are presented in Section 5.

3.2 Source Schema and Data Warehouse Schema: the Logical Level

Besides the conceptual level of the representation, our approach requires that the various sources and the Data Warehouse itself are described at the logical level (here the term “logical” refers to the database terminology).

The logical content of each source S, called the Source Schema, is provided in terms of a set of definitions of relations, each one expressed in terms of a query over the Source Model of S. The logical content of a source represents the structure of data expressed in terms of a logical data model, in this case the Relational Model. In our framework, such a structure is provided by specifying the connection with the conceptual representation of the source. In other words, the logical content of a source S, or of a portion thereof, is described in terms of a view over the Source Model associated with S (and, therefore, of the Conceptual Data Warehouse Model). We envision the use of suitable mechanisms, called wrappers, to map physical structures to logical structures (see Figure 2).

The logical content of the materialized views constituting the Data Warehouse, called the Data Warehouse Schema, is provided in terms of a set of definitions of relations, each one expressed in terms of a query over the Conceptual Data Warehouse Model. Similarly to the case of the sources, each portion of the Data Warehouse Schema is described in terms of a view over the Conceptual Data Warehouse Model. The way in which a view is actually materialized starting from the data in the sources is specified by means of suitable mechanisms, called mediators (see Figure 2).

The formalism for expressing both the Source and the Data Warehouse Schema is described in Section 6, while the reasoning techniques on the logical level are discussed in Section 7.

4 The Formalism for the Conceptual Level

The formalism we use for the conceptual level belongs to the family of *Description Logics*, introduced and studied in the field of Knowledge Representation (see e.g. [Bor95, DLNS96]). Generally speaking, Description Logics are based on objects, classes (or concepts) and relationships. In many fields of Computer Science we find formalisms for the representation of objects and classes. A *class* (or *concept*) denotes a subset of the domain of discourse, and a class-based representation formalism allows one to express several kinds of relationships and constraints (e.g. subclass constraints) holding among classes. Moreover, class-based formalisms aim at taking advantage of the class structure in order to provide various information, such as whether an element belongs to a class, whether a class is a subclass of another class, and more generally, whether a given constraint holds for a given set of classes.

In traditional Description Logics, structured knowledge is described by means of so called *concepts* and *roles*, which denote unary and binary predicates, respectively. Starting from a set of atomic symbols one can build complex concept and role expressions by applying suitable constructors which characterize a Description Logic. Formally, concepts are interpreted as subsets of a domain and roles as binary relations over that domain, and all constructors are equipped with a precise set-theoretic semantics. The most common constructors include boolean operators on concepts, and quantification over roles. For example, the concept $\texttt{Person} \sqcap \forall\texttt{child}.\texttt{Male}$, denotes the set of individuals that are instances of the concept `Person` and are connected through the role `child` only to instances of the concept `Male`, while the concept $\exists\texttt{child}$ denotes all individuals that are connected through role `child` to some individual. Further constructors that have been considered important include more general forms of quantification, number restrictions, which allow one to state limits on the number of connections that an individual may have via a certain role, and constructors on roles, such as intersection, concatenation and inverse. A Description Logic knowledge base, expressing the intensional knowledge about the modeled domain, is built by stating inclusion and/or equality assertions between concepts, which have to be satisfied by the models of the knowledge base. The assertions are used to specify necessary and/or necessary and sufficient conditions for individuals to be instances of certain concepts. Reasoning on such knowledge bases includes the detection of inconsistencies in the knowledge base itself, determining whether a concept can be populated in a model of the knowledge base, and checking subsumption, i.e. whether all instances of a concept are necessarily also instances of another concept in all models of the knowledge base.

For the conceptual level of a Data Warehouse, we propose here a new Description Logic, called $\mathcal{DLR}$, which includes *concepts* and *n-ary relations*. $\mathcal{DLR}$ is inspired by the languages introduced in [CDGL97a, CDGL95, DGL95, DGL94b, CL93], where the notion of n-ary relation was already present. We believe that $\mathcal{DLR}$ is a natural extension of Description Logics towards n-ary relations, that are extremely important in our context.

4.1 Syntax of $\mathcal{DLR}$

We deal with an alphabet $\Sigma = \Sigma_{\mathcal{C}} \dot\cup \Sigma_{\mathcal{R}} \dot\cup \Sigma_{\mathcal{U}}$ in which:

1. $\Sigma_{\mathcal{R}}$ is a finite set of *atomic relations* (ranged over by $\mathbf{P}$);
2. $\Sigma_{\mathcal{U}}$ is a finite set of *attribute names* (ranged over by U);
3. $\Sigma_{\mathcal{C}}$ is a finite set of *atomic concepts* (ranged over by A).

We consider also a finite set $\Sigma_{\mathcal{D}}$ of *domains* (ranged over by D), and each domain $D \in \Sigma_{\mathcal{D}}$ has an associated set of *domain values* (ranged over by v).

Each atomic relation $\mathbf{P}$ has an associated *signature* $\tau = \langle U_1 : t_1, \ldots, U_n : t_n \rangle$, where

- $n \geq 2$ is the *arity* of $\mathbf{P}$;
- U_i ($i \in \{1, \ldots, n\}$) is an attribute name denoting the i-th component of the relation;
- t_i ($i \in \{1, \ldots, n\}$) is the *type* of the i-th component which can be either *obj* (denoting objects) or *val* (denoting values).

We say that two signatures $\tau = \langle U_1 : t_1, \ldots, U_n : t_n \rangle$ and $\tau' = \langle U'_1 : t'_1, \ldots, U'_{n'} : t'_{n'} \rangle$ are *compatible* if $n = n'$ and $t_i = t'_i$ for each $i \in \{1, \ldots, n\}$, i.e. τ and τ' may differ in the attribute names, but not in the arity or the types of the components.

We assume that for each equivalence class $[\tau]$ formed by all signatures τ' that are compatible with τ, $\Sigma_{\mathcal{R}}$ contains a special atomic relation $\top_{[\tau]}$ of signature τ. Similarly, we assume that $\Sigma_{\mathcal{C}}$ contains a special atomic concept $\top_{\mathcal{C}}$.

We use $\mathbf{R}^{(\tau)}$, C, and V to denote, respectively, arbitrary relations of signature τ, arbitrary concepts, and domains, which are built starting from atomic relations, atomic concepts, attributes, domains, and domain values by applying *constructors* according to the following syntax:

$$\begin{array}{rcl}
\mathbf{R}^{(\tau)} & ::= & \mathbf{P}^{(\tau)} \mid (U_o^{(\tau)} : C) \mid (U_v^{(\tau)} : V) \mid \neg \mathbf{R}^{(\tau)} \mid \mathbf{R}_1^{(\tau)} \sqcap \mathbf{R}_2^{(\tau)} \mid \delta_\tau(\mathbf{R}^{(\tau')}) \\
C & ::= & A \mid \neg C \mid C_1 \sqcap C_2 \mid \forall \mathbf{R}_1^{(\tau)}[U_o].\mathbf{R}_2^{(\tau)} \mid \exists^{\geq k} \mathbf{R}_1^{(\tau)}[U_o].\mathbf{R}_2^{(\tau)} \\
V & ::= & D \mid \{v_1, \ldots, v_h\}
\end{array}$$

where U_o is an attribute in τ denoting a component of type *obj*, U_v is an attribute in τ denoting a component of type *val*, τ and τ' in $\delta_\tau(\mathbf{R}^{(\tau')})$ are *compatible*, and each v_i in $\{v_1, \ldots, v_h\}$ is a value associated with some domain D_i in $\Sigma_{\mathcal{D}}$. We omit the signature of a relation whenever it is apparent from the context.

Before providing the formal semantics we briefly discuss the intuitive meaning of the introduced constructors:

- $\mathbf{R}_1^{(\tau)} \sqcap \mathbf{R}_2^{(\tau)}$ denotes *conjunction of relations* with identical signature.
- $(U_o^{(\tau)} : C)$ denotes a relation of signature τ whose component U_o is an instance of C (similarly for $(U_v^{(\tau)} : V)$). These are used essentially to impose conditions on *single components of relations*.
- $\neg \mathbf{R}^{(\tau)}$ is used essentially together with conjunction to denote *difference of relations*.
- $\delta_\tau(\mathbf{R}^{(\tau')})$ denotes *signature renaming* and is used for the construction of complex relations out of relations with compatible (but not identical) signatures.
- $\neg C$ and $C_1 \sqcap C_2$ denote *negation* and *conjunction of concepts*. We consider *disjunction* of concepts $C_1 \sqcup C_2$ as an abbreviation for $\neg(\neg C_1 \sqcap \neg C_2)$.
- $\forall \mathbf{R}_1^{(\tau)}[U_o].\mathbf{R}_2^{(\tau)}$ denotes *universal quantification* over relations and is used to impose conditions on the set of all tuples of which an object is the U_o component. Notice that this constructor cannot be used with attributes denoting components of type *val*.
- $\exists^{\geq k} \mathbf{R}_1^{(\tau)}[U_o].\mathbf{R}_2^{(\tau)}$ denotes *qualified number restrictions* over relations and imposes a limit on the minimum number of tuples of a relation in which an object is the U_o component. In particular, when k is equal to 1 it can be used to denote existence dependencies and, together with negation, functional dependencies.

4.2 Semantics of $\mathcal{DLR}$

We give the semantics of our language by providing a translation $fol(\cdot)$ of concepts and relations to open formulae of first order logic with equality, and relying on first order interpretations.

We use lambda notation for open formulae, hence interpreting $E = \lambda\vec{\mathbf{x}}.\gamma(\vec{\mathbf{x}})$ as a predicate of arity n, where n is the number of variables of $\vec{\mathbf{x}}$. By the application $E(\vec{\mathbf{y}})$ we denote $\gamma(\vec{\mathbf{y}})$.

Let $\mathbf{P}, \mathbf{R}, \mathbf{R}_1, \mathbf{R}_2$ be of signature τ. The semantics of the constructors of $\mathcal{DLR}$ is as follows:

$$
\begin{array}{rcl}
fol(\mathbf{P}^{(\tau)}) & = & \lambda\vec{\mathbf{x}}.\top_{[\tau]}(\vec{\mathbf{x}}) \wedge \mathbf{P}^{(\tau)}(\vec{\mathbf{x}}) \\
fol((U_i^{(\tau)}\colon C)) & = & \lambda\vec{\mathbf{x}}.\top_{[\tau]}(\vec{\mathbf{x}}) \wedge fol(C)(x_i) \\
fol((U_i^{(\tau)}\colon V)) & = & \lambda\vec{\mathbf{x}}.\top_{[\tau]}(\vec{\mathbf{x}}) \wedge fol(V)(x_i) \\
fol(\neg\mathbf{R}^{(\tau)}) & = & \lambda\vec{\mathbf{x}}.\top_{[\tau]}(\vec{\mathbf{x}}) \wedge \neg fol(\mathbf{R}^{(\tau)})(\vec{\mathbf{x}}) \\
fol(\mathbf{R}_1^{(\tau)} \sqcap \mathbf{R}_2^{(\tau)}) & = & \lambda\vec{\mathbf{x}}.fol(\mathbf{R}_1^{(\tau)})(\vec{\mathbf{x}}) \wedge fol(\mathbf{R}_2^{(\tau)})(\vec{\mathbf{x}}) \\
fol(\delta_{\tau'}(\mathbf{R}^{(\tau)})) & = & fol(\mathbf{R}^{(\tau)}) \\[1ex]
fol(A) & = & \lambda x.\top_{\mathcal{C}}(x) \wedge A(x) \\
fol(\neg C) & = & \lambda x.\top_{\mathcal{C}}(x) \wedge \neg fol(C)(x) \\
fol(C_1 \sqcap C_2) & = & \lambda x.fol(C_1)(x) \wedge fol(C_2)(x) \\
fol(\forall\mathbf{R}_1[U_i].\mathbf{R}_2) & = & \lambda x.\top_{\mathcal{C}}(x) \wedge \forall\vec{\mathbf{x}}.fol(\mathbf{R}_1)(\vec{\mathbf{x}}) \wedge x = x_i \supset fol(\mathbf{R}_2)(\vec{\mathbf{x}}) \\
fol(\exists^{\geq k}\mathbf{R}_1[U_i].\mathbf{R}_2) & = & \lambda x.\top_{\mathcal{C}}(x) \wedge \exists^{\geq k}\vec{\mathbf{x}}.x = x_i \wedge fol(\mathbf{R}_1)(\vec{\mathbf{x}}) \wedge fol(\mathbf{R}_2)(\vec{\mathbf{x}}) \\[1ex]
fol(D) & = & \lambda x.D(x) \\
fol(\{v_1, \ldots, v_h\}) & = & \lambda x.(x = v_1 \vee \cdots \vee x = v_h)
\end{array}
$$

Given a first order interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$, a formula defined by the lambda expression $\lambda\vec{\mathbf{x}}.\gamma(\vec{\mathbf{x}})$ is interpreted as

$$(\lambda\vec{\mathbf{x}}.\gamma(\vec{\mathbf{x}}))^{\mathcal{I}} \;=\; \{\vec{\mathbf{d}} \in \Delta^{\mathcal{I}} \mid \gamma^{\mathcal{I},[\vec{\mathbf{x}}/\vec{\mathbf{d}}]} \text{ is true}\}$$

Hence the set $C^{\mathcal{I}}$ of instances of C in $\mathcal{I}$ is $fol(C)^{\mathcal{I}}$. Similarly, the set $\mathbf{R}^{\mathcal{I}}$ of tuples that are instances of $\mathbf{R}$ is $fol(\mathbf{R})^{\mathcal{I}}$.

Notice that in the semantics we have enforced that the extension of every relation of signature τ is contained in the extension of $\top_{[\tau]}$. However the extension of $\top_{[\tau]}$ is not forced to be the Cartesian product of the extensions of the single components in τ. Hence negated relations essentially are able to capture difference of relations, since the extension of $\neg\mathbf{R}^{(\tau)}$ is obtained as the difference between the extension of $\top_{[\tau]}$ and the extension of $\mathbf{R}^{(\tau)}$.

In order to take into account concrete domains (such as strings, integers, reals, ...) we restrict our interest to first-order interpretation of a special form, called *admissible interpretations*, in which each domain and each domain value have a fixed extension. Formally, we define a function $dom(\cdot)$ which associates a concrete domain $dom(D)$ with each domain name $D \in \Sigma_{\mathcal{D}}$, and define

$$\Delta_{\mathcal{D}} \;=\; \bigcup_{D \in \Sigma_{\mathcal{D}}} dom(D).$$

Moreover, we define a function $\alpha(\cdot)$ which associates each domain value v of the domain D with a distinct element of the concrete domain $domD$, i.e. $\alpha(v) \in dom(D)$, and $\alpha(v_1) \neq \alpha(v_2)$ if $v_1 \neq v_2$.

The first order interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over the alphabet Σ is *admissible* if the following conditions are satisfied:

1. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathcal{C}} \,\dot{\cup}\, \Delta_{\mathcal{D}}$;
2. $(\lambda x.D(x))^{\mathcal{I}} = dom(D)$ for each $D \in \Sigma_{\mathcal{D}}$;
3. $v^{\mathcal{I}} = \alpha(v)$ for each value of each $D \in \Sigma_{\mathcal{D}}$;
4. ${\top_{\mathcal{C}}}^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathcal{C}}$;
5. $\top^{\mathcal{I}}_{[\langle U_1:t_1,\ldots,U_n:t_n\rangle]} \subseteq \Delta_1 \times \cdots \times \Delta_n, \qquad \Delta_i = \begin{cases} \Delta^{\mathcal{I}}_{\mathcal{C}} & \text{if } t_i = obj \\ \Delta_{\mathcal{D}} & \text{if } t_i = val \end{cases} \qquad$ for each $\top_{[\tau]} \in \Sigma_{\mathcal{R}}$.

In the following we will implicitly consider only admissible interpretations.

4.3 The Enterprise and Source Models

As we said before, we consider two kinds of conceptual models in source integration, namely, the Enterprise Model, and the Source Models, one for each relevant source. In this subsection, we use the term *conceptual model* to denote either the Enterprise Model or one of the Source Models.

A $\mathcal{DLR}$ *conceptual model* $\mathcal{M}$ over the alphabet Σ and domains $\Sigma_\mathcal{D}$ is constituted by a finite set of *assertions*, of the form

$$L_i \sqsubseteq L_j \qquad L_i \not\sqsubseteq L_j \qquad L_i \equiv L_j \qquad L_i \not\equiv L_j$$

where L_1 and L_2 are either two relations with compatible signatures or two concepts over $\Sigma \cup \Sigma_\mathcal{D}$.

The semantics of assertions is also given in terms of $fol(\cdot)$, by deriving the corresponding first order closed formulae:

$$\begin{array}{rcl}
fol(\mathbf{R}_i^{(\tau)} \sqsubseteq \mathbf{R}_j^{(\tau')}) &=& \forall \vec{\mathbf{x}}. fol(\mathbf{R}_i^{(\tau)})(\vec{\mathbf{x}}) \supset fol(\mathbf{R}_j^{(\tau')})(\vec{\mathbf{x}}) \\
fol(\mathbf{R}_i^{(\tau)} \not\sqsubseteq \mathbf{R}_j^{(\tau')}) &=& \neg fol(\mathbf{R}_i^{(\tau)} \sqsubseteq \mathbf{R}_j^{(\tau')}) \\
fol(\mathbf{R}_i^{(\tau)} \equiv \mathbf{R}_j^{(\tau')}) &=& \forall \vec{\mathbf{x}}. fol(\mathbf{R}_i^{(\tau)})(\vec{\mathbf{x}}) \equiv fol(\mathbf{R}_j^{(\tau')})(\vec{\mathbf{x}}) \\
fol(\mathbf{R}_i^{(\tau)} \not\equiv \mathbf{R}_j^{(\tau')}) &=& \neg fol(\mathbf{R}_i^{(\tau)} \equiv \mathbf{R}_j^{(\tau')}) \\
fol(C_i \sqsubseteq C_j) &=& \forall x. fol(C_i)(x) \supset fol(C_j)(x) \\
fol(C_i \not\sqsubseteq C_j) &=& \neg fol(C_i \sqsubseteq C_j) \\
fol(C_i \equiv C_j) &=& \forall x. fol(C_i)(x) \equiv fol(C_j)(x) \\
fol(C_i \not\equiv C_j) &=& \neg fol(C_i \equiv C_j)
\end{array}$$

An interpretation $\mathcal{I}$ *satisfies* an assertion $L_1 \propto L_2$ if and only if $fol(L_1 \propto L_2)$ evaluates to true in $\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model* of a conceptual model $\mathcal{M}$ if and only if:

- $\mathcal{I}$ is an admissible interpretation;
- $\mathcal{I}$ satisfies all assertions in $\mathcal{M}$.

Let us consider an example, which we will extend in the following sections. We would like to model an enterprise and two sources containing information about contracts between clients and departments for services, and about registration of clients at departments. Source 1 contains information about clients registered at public-relations departments. Source 2 contains information about contracts and complete information about services. Such situation can be represented by means of one Enterprise Model $\mathcal{M}_0$, and two Source Models $\mathcal{M}_1$ and $\mathcal{M}_2$. The alphabets $\Sigma_0, \Sigma_1, \Sigma_2$ of the three models are defined as follows:

- $\Sigma_0 = \{\mathtt{CONTRACT}_0, \mathtt{REG\text{-}AT}_0, \mathtt{Client}_0, \mathtt{Service}_0, \mathtt{Department}_0, \mathtt{PrDept}_0\}$
- $\Sigma_1 = \{\mathtt{REG\text{-}AT}_1, \mathtt{PROMOTION}_1, \mathtt{Client}_1, \mathtt{Department}_1, \mathtt{LOCATION}_1\}$
- $\Sigma_2 = \{\mathtt{CONTRACT}_2, \mathtt{Client}_2, \mathtt{Department}_2, \mathtt{Service}_2\}$

The various relations in the alphabets have the following signatures:

$\mathtt{CONTRACT}_0$, $\mathtt{CONTRACT}_2$, have signature $\tau = \langle \mathtt{cli}\colon obj, \mathtt{dep}\colon obj, \mathtt{ser}\colon obj\rangle$;
$\mathtt{REG\text{-}AT}_0$, $\mathtt{REG\text{-}AT}_1$, $\mathtt{PROMOTION}_1$, have signature $\tau' = \langle \mathtt{cli}\colon obj, \mathtt{dep}\colon obj\rangle$;
$\mathtt{LOCATION}_1$ has signature $\tau'' = \langle \mathtt{dep}\colon obj, \mathtt{loc}\colon val\rangle$.

Finally, the assertions defining the three models are as follows:

- $\mathcal{M}_0$ contains the assertions:

$$\begin{array}{rcl}
\mathtt{CONTRACT}_0 &\sqsubseteq& (\mathtt{cli}\colon \mathtt{Client}_0) \sqcap (\mathtt{dep}\colon \mathtt{Department}_0) \sqcap (\mathtt{ser}\colon \mathtt{Service}_0) \\
\mathtt{REG\text{-}AT}_0 &\sqsubseteq& (\mathtt{cli}\colon \mathtt{Client}_0) \sqcap (\mathtt{dep}\colon \mathtt{Department}_0) \\
\mathtt{PrDept}_0 &\sqsubseteq& \mathtt{Department}_0
\end{array}$$

- $\mathcal{M}_1$ contains the assertions:

$$
\begin{array}{rcl}
\texttt{REG-AT}_1 & \sqsubseteq & (\texttt{cli}\colon\texttt{Client}_1) \sqcap (\texttt{dep}\colon\texttt{Department}_1) \\
\texttt{PROMOTION}_1 & \sqsubseteq & \texttt{REG-AT}_1 \\
\texttt{LOCATION}_1 & \sqsubseteq & (\texttt{dep}\colon\texttt{Department}_1) \sqcap (\texttt{loc}\colon\texttt{String}) \\
\texttt{Department}_1 & \sqsubseteq & \exists^{\leq 1}\texttt{LOCATION}_1[\texttt{dep}].\top_{[\mathcal{T}'']}
\end{array}
$$

- $\mathcal{M}_2$ contains the assertions:

$$
\texttt{CONTRACT}_2 \sqsubseteq (\texttt{cli}\colon\texttt{Client}_2) \sqcap (\texttt{dep}\colon\texttt{Department}_2) \sqcap (\texttt{ser}\colon\texttt{Service}_2)
$$

4.4 Relationship with other Data Models

In this subsection, we discuss the relationships between the formalism we introduced and other popular data models. In particular, we show how to express in our framework the Entity-Relationship Model, the Relational Model, and the Object-Oriented Data Model.

4.4.1 Relationship with the Entity-Relationship Model

We now show that the semantics of the *Entity-Relationship* (ER) Model can be captured by the Description Logic $\mathcal{DLR}$. The work reported in [CLN94] defines a translation from ER schemas to the Description Logic $\mathcal{ALCQI}$, which is a restriction of the logic $\mathcal{DLR}$, and a correspondence is established between legal database states and models of the derived knowledge base.

Based on such a translation, in the following we define a transformation of ER schemas into $\mathcal{DLR}$ knowledge bases, whose correctness follows straightforwardly from the results reported in [CLN94].

The Entity-Relationship Model The basic elements of the ER Model are entities, relationships, and attributes. An *entity set* (or simply *entity*) denotes a set of objects, called its *instances*, that have common properties. Elementary properties are modeled through *attributes*, whose values belong to one of several predefined domains, such as `Integer`, `String`, or `Boolean`. Properties that are due to relations to other entities are modeled through the participation of the entity in relationships. A *relationship set* (or simply *relation*) denotes a set of tuples (also called its instances), each of which represents an association among a different combination of instances of the entities that participate in the relationship. Since each entity can participate in a relationship more than once, the notion of *ER-role* is introduced, which represents such a participation and to which a unique name is assigned. The *arity* of a relationship is the number of its ER-roles. *Cardinality constraints* can be attached to an ER-role in order to restrict the number of times each instance of an entity is allowed to participate via that ER-role in instances of the relationship. Such constraints can be used to specify both existence dependencies and functionality of relations [CK86]. They are often used only in a restricted form, where the minimum cardinality is either 0 or 1 and the maximum cardinality is either 1 or ∞. Cardinality constraints in the form considered here have been introduced already in [Abr74] and subsequently studied in [GM84, LN90, Fer91, YPS94, Tha92, CL94b, CL94a]. Additionally, so called *ISA* relations are used to represent inclusion assertions between entities and between relationships, and therefore the inheritance of properties from a more general entity to a more specific one.

We define now in a more formal way syntax and semantics of ER schemas.

Syntax of Entity-Relationship Schemas An *ER schema* $\mathcal{S}$ is constructed starting from pairwise disjoint sets of entity symbols, relationship symbols, ER-role symbols, attribute symbols, and domain symbols. Each domain symbol D has an associated predefined basic domain $D^{\mathcal{B}_\mathcal{D}}$, and we assume the basic domains to be pairwise disjoint. For each entity symbol a set of attribute symbols is defined, and to each such attribute a unique domain symbol is associated. Each relationship symbol of arity k has k associated ER-role symbols, each with an associated entity symbol, and defines a relationship between these entities. We assume that each ER-role symbol belongs to a unique relationship, thus determining also a unique entity. The cardinality constraints are represented by two functions $cmin_\mathcal{S}$, from ER-role symbols to nonnegative integers, and $cmax_{\mathcal{S}}$, from ER-role symbols to positive integers union the special symbol ∞. ISA relations between entities and between relationships are modeled by means of a binary relation $\preceq_{\mathcal{S}}$. We do not need to make any special assumption on the form of $\preceq_{\mathcal{S}}$, such as acyclicity or injectivity.

In the commonly accepted graphical ER notation, entities are represented as boxes, whereas relationships are represented as diamonds. An attribute is shown as a circle attached to the entity for which it is defined. ER-roles are graphically depicted by connecting the relationship to the participating entities and labeling the connection with the associated cardinality constraints. An *ISA* relation between two entities is denoted by an arrow from the more specific to the more general entity (analogously for ISA relations between two relationships).

Semantics of Entity-Relationship Schemas The semantics of an ER schema can be given by specifying which database states are consistent with the information structure represented by the schema. Formally, a database state $\mathcal{B}$ corresponding to an ER schema $\mathcal{S}$ is constituted by a nonempty *finite* set $\Delta^{\mathcal{B}}$, assumed to be disjoint from all basic domains, and a function $\cdot^{\mathcal{B}}$ that maps

- every domain symbol D to the corresponding basic domain $D^{\mathcal{B}_{\mathcal{D}}}$,
- every entity E to a subset $E^{\mathcal{B}}$ of $\Delta^{\mathcal{B}}$,
- every attribute A to a set $A^{\mathcal{B}} \subseteq \Delta^{\mathcal{B}} \times \bigcup_{D \in \mathcal{D}_{\mathcal{S}}} D^{\mathcal{B}_{\mathcal{D}}}$, and
- every relationship R to a set $R^{\mathcal{B}}$ of labeled tuples over $\Delta^{\mathcal{B}}$.

A *labeled tuple* over a domain $\Delta^{\mathcal{B}}$ is a function from a set of ER-roles to $\Delta^{\mathcal{B}}$. The labeled tuple T that maps ER-role U_i to o_i, for $i \in \{1, \ldots, n\}$, is denoted $\langle U_1 \colon o_1, \ldots, U_n \colon o_n \rangle$. We also write $T[U_i]$ to denote o_i, and call it the *U_i-component of T*. The elements of $E^{\mathcal{B}}$, $A^{\mathcal{B}}$, and $R^{\mathcal{B}}$ are called *instances* of E, A, and R respectively.

A database state is considered acceptable if it satisfies all integrity constraints that are part of the schema. This is captured by the notion of legal database state. A database state $\mathcal{B}$ is *legal for* an ER schema $\mathcal{S}$, if it satisfies the following conditions:

- For each pair of entities E_1, E_2 with $E_1 \preceq_{\mathcal{S}} E_2$, it holds that $E_1^{\mathcal{B}} \subseteq E_2^{\mathcal{B}}$.
- For each pair of relationships R_1, R_2 with $R_1 \preceq_{\mathcal{S}} R_2$, it holds that $R_1^{\mathcal{B}} \subseteq R_2^{\mathcal{B}}$.
- For each entity E, if E has an attribute A with domain D, then for each instance $e \in E^{\mathcal{B}}$ there is exactly one element $a \in A^{\mathcal{B}}$ with e as first component, and the second component of a is an element of $D^{\mathcal{B}_{\mathcal{D}}}$.
- For each relationship R of arity k between entities $E_1, \ldots, E_n$, to which R is connected by means of ER-roles $U_1, \ldots, U_n$ respectively, all instances of R are of the form $\langle U_1 \colon e_1, \ldots, U_n \colon e_n \rangle$, where $e_i \in E_i^{\mathcal{B}}$, $i \in \{1, \ldots, n\}$.
- For each ER-role U associated to relationship R and entity E, and for each instance e of E, it holds that
$$cmin_{\mathcal{S}}(U) \leq \sharp\{r \in R^{\mathcal{B}} \mid r[U] = e\} \leq cmax_{\mathcal{S}}(U).$$

Transforming Entity-Relationship Schemas into $\mathcal{DLR}$ Knowledge Bases We now show that the semantics of the ER Model can be captured in the Description Logic $\mathcal{DLR}$ by defining a translation ϕ from ER schemas to $\mathcal{DLR}$ knowledge bases, and then establishing a correspondence between legal database states and models of the derived knowledge base.

The knowledge base $\phi(\mathcal{S})$ derived from an ER schema $\mathcal{S}$ is defined as follows: The set of atomic concept of $\phi(\mathcal{S})$ consists of the set of entity symbols in $\mathcal{S}$. The set of atomic relations of $\phi(\mathcal{S})$ consists of the set of relationship and attribute symbols in $\mathcal{S}$. A relationship R with ER-roles $U_1, \ldots, U_n$ in $\mathcal{S}$ has signature $\tau_R = \langle U_1 \colon obj, \ldots, U_n \colon obj \rangle$ in ERS. All relations corresponding to attributes have the same signature $\tau_{att} = \langle \texttt{from} \colon obj, \texttt{to} \colon val \rangle$ in ERS. The set of domain names of $\phi(\mathcal{S})$ corresponds to the set of domain names of $\mathcal{S}$.

The set of assertions of $\phi(\mathcal{S})$ consists of the following elements:

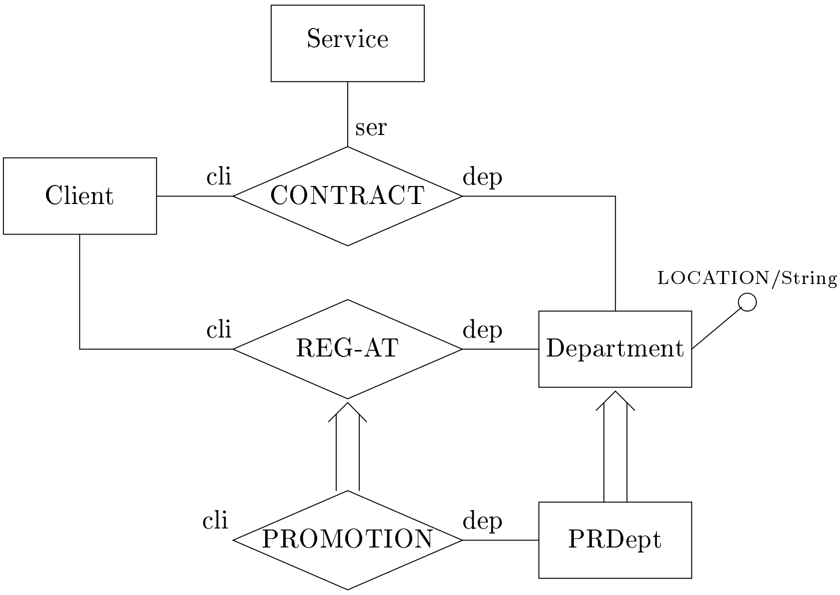

Figure 3: Example of an ER schema

- For each pair of entities E_1, E_2 such that $E_1 \preceq_{\mathcal{S}} E_2$, the assertion

$$E_1 \sqsubseteq E_2$$

- For each pair of relationships R_1, R_2 such that $R_1 \preceq_{\mathcal{S}} R_2$, the assertion

$$R_1 \sqsubseteq R_2$$

- For each attribute A with domain D of an entity E,

$$E \sqsubseteq \forall A[\mathtt{from}].(\mathtt{to}^{(\tau_{att})}\colon D) \sqcap \exists^{=1} A[\mathtt{from}].\top_{[\tau_{att}]}$$

- For each relationship R of arity n between entities $E_1, \ldots, E_n$, to which R is connected by means of ER-roles $U_1, \ldots, U_n$ respectively, the assertion

$$R \sqsubseteq (U_1^{(\tau_R)}\colon E_1) \sqcap \cdots \sqcap (U_k^{(\tau_R)}\colon E_n)$$

- For each ER-role U associated with relationship R and entity E,
  1. if $m = cmin_{\mathcal{S}}(U) \neq 0$, the assertion

  $$E \sqsubseteq \exists^{\geq m} R[U].\top_{[\tau_R]}$$

  2. if $n = cmax_{\mathcal{S}}(U) \neq \infty$, the assertion

  $$E \sqsubseteq \exists^{\leq n} R[U].\top_{[\tau_R]}$$

Based on the results presented in [CLN94], we are able to prove the correctness of the translation presented above. More specifically, let $\mathcal{S}$ be an ER schema. Then, there is a one-to-one correspondence between legal database states of $\mathcal{S}$ and models of the $\mathcal{DLR}$ knowledge base $\phi(\mathcal{S})$. For example, an entity E can be populated in a legal database state for $\mathcal{S}$ if and only if $\phi(\mathcal{S})$ admits a model in which E has a non-empty extension. This allows us to exploit reasoning techniques developed for the logic $\mathcal{DLR}$ in order to reason on ER schemas.

We present a simple example of the translation of an ER schema to a $\mathcal{DLR}$ knowledge base. By applying the translation presented above to the ER schema shown in Figure 3, we obtain the following $\mathcal{DLR}$ knowledge base:

$$\begin{array}{rcl}
\texttt{CONTRACT} & \sqsubseteq & (\texttt{cli}:\texttt{Client}) \sqcap (\texttt{ser}:\texttt{Service}) \sqcap (\texttt{dep}:\texttt{Department}) \\
\texttt{REG-AT} & \sqsubseteq & (\texttt{cli}:\texttt{Client}) \sqcap (\texttt{dep}:\texttt{Department}) \\
\texttt{PROMOTION} & \sqsubseteq & \texttt{REG-AT} \sqcap (\texttt{dep}:\texttt{PrDept}) \\
\texttt{Department} & \sqsubseteq & \forall \texttt{LOCATION[from]}.(\texttt{to}^{(\tau_{att})}:\texttt{String}) \sqcap \exists^{=1}\texttt{LOCATION[from]}.\top_{[\tau_{att}]} \\
\texttt{PrDept} & \sqsubseteq & \texttt{Department}
\end{array}$$

Additions to the Entity-Relationship Model The ER Model does not provide several features and modeling primitives which would prove useful in order to represent complex dependencies between data. The richness of constructs that is typical of Description Logics, and the correspondence between the two formalisms established in the previous section makes it however possible to add such constructs to the basic model and take them fully into account when reasoning on a schema. We provide now several meaningful examples of possible additions to the basic ER Model that arise as a natural consequence of the correspondence with the Description Logic $\mathcal{DLR}$.

- *Arbitrary boolean constructs on entities.* The only direct relationship between entities that can be expressed in the basic ER Model is the ISA relation. A common extension is by so called *generalization hierarchies* (see e.g. [BCN92]), which allow one to express that the extension of an entity should be the disjoint union of the extensions of other entities. Such construct can easily be translated by making use of union and negation of $\mathcal{DLR}$.
- *Refinement of properties along an ISA hierarchy.* Another important extension that should be considered is the possibility to specify more complex forms of refinement of properties of entities along ISA hierarchies, than the mere addition of attributes. This is already an essential feature of the more recent Object-Oriented Models. In particular, cardinality constraints could be refined by restricting the range of values, and the participation in relationships can be restricted. One may require for specific instances of an entity that the objects they are related to via a certain relationship belong to a more specific entity than the one directly associated to the ER-role. Such forms of constraints can be naturally expressed in $\mathcal{DLR}$ by making use of universal quantification over relations.
- *Definitions of classes by means of complex properties.* In the ER Model (and more generally in Semantic Data Models) one can specify only necessary conditions that the instances of entities (or more generally classes) must satisfy. This means that in a database that conforms to the schema one cannot deduce that a certain object is an instance of an entity unless this fact is explicitly stated. By using the different types of assertions of $\mathcal{DLR}$, necessary and sufficient (and even just sufficient) conditions can be easily imposed and become part of the schema.

4.4.2 Relationship with the Relational Model

There is also a tight correspondence between the Description Logic $\mathcal{DLR}$ and the Relational Data Model. In fact, relations in the Relational Model can be represented in terms of $\mathcal{DLR}$-relations, while the constructors of $\mathcal{DLR}$ can be used to represent dependencies in the Relational Model. In particular:

- unary functional dependencies can be expressed by qualified number restrictions in $\mathcal{DLR}$;
- unary inclusion dependencies and inclusions between relations can be expressed in terms of inclusion assertions in $\mathcal{DLR}$.

4.4.3 Relationship with Object-Oriented Models

In [CLN94] a simple object-oriented language in the style of most popular models featuring complex objects and object identity is defined, and a relationship between Description Logics and this model is established by providing a mapping from Object-Oriented schemas into Description Logic knowledge bases[2]. Since the Description Logic used in [CLN94] is subsumed by the logic $\mathcal{DLR}$, such a mapping can also be used to reformulate reasoning on an object-oriented schema in terms of reasoning on the corresponding $\mathcal{DLR}$ knowledge base.

Moreover, the constructors of $\mathcal{DLR}$ can again be used to extend the expressiveness of the modeling language through e.g. cardinality restrictions and inverse of attributes.

5 The Conceptual Data Warehouse Model

The Conceptual Data Warehouse Model provides a conceptual representation of the enterprise and the various data sources. Formally, we consider a Conceptual Data Warehouse Model to be a set of conceptual models $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m$, where $\mathcal{M}_0$ is the Enterprise Model, and each $\mathcal{M}_i$, with $i \in \{1, \ldots, m\}$, is the Source Model for source S_i. Each conceptual model $\mathcal{M}_i$ ($i \in \{0, \ldots, m\}$) is over an alphabet Σ_i and a set $\Sigma_{\mathcal{D}}$ of common domains.

Inclusion assertions in a conceptual model $\mathcal{M}_i$ express knowledge on the relations and concepts in $\mathcal{M}_i$. For this reason such assertions are also called *intramodel assertions*. Additionally, to specify knowledge on the conceptual interrelationships among the various sources and/or the enterprise, we introduce special assertions called intermodel assertions [CL93]. An *intermodel assertion* over Σ_i, Σ_j ($i \neq j$) is an assertion of one of the following forms

$$\begin{array}{lllllllllll} L_i & \sqsubseteq_{ext} & L_j & \quad & L_i & \not\sqsubseteq_{ext} & L_j & \quad L_i & \equiv_{ext} & L_j & \quad L_i \not\equiv_{ext} L_j \\ L_i & \sqsubseteq_{int} & L_j & \quad & L_i & \not\sqsubseteq_{int} & L_j & \quad L_i & \equiv_{int} & L_j & \quad L_i \not\equiv_{int} L_j \end{array}$$

in which L_i and L_j are either two relations with compatible signatures or two concepts over Σ_i and Σ_j respectively.

Intermodel assertions are interpreted as closed first order formulae, by extending the mapping $fol(\cdot)$ as follows.

- If $\mathbf{R}_i^{(\tau)}$ and $\mathbf{R}_j^{(\tau')}$ are two relations over the alphabets Σ_i and Σ_j respectively, with τ, τ' compatible, then

$$\begin{array}{rcl}
fol(\mathbf{R}_i^{(\tau)} \sqsubseteq_{ext} \mathbf{R}_j^{(\tau')}) & = & \forall \vec{\mathbf{x}}. fol(\mathbf{R}_i^{(\tau)})(\vec{\mathbf{x}}) \supset fol(\mathbf{R}_j^{(\tau')})(\vec{\mathbf{x}}) \\
fol(\mathbf{R}_i^{(\tau)} \not\sqsubseteq_{ext} \mathbf{R}_j^{(\tau')}) & = & \neg fol(\mathbf{R}_i^{(\tau)} \sqsubseteq_{ext} \mathbf{R}_j^{(\tau')}) \\
fol(\mathbf{R}_i^{(\tau)} \equiv_{ext} \mathbf{R}_j^{(\tau')}) & = & \forall \vec{\mathbf{x}}. fol(\mathbf{R}_i^{(\tau)})(\vec{\mathbf{x}}) \equiv fol(\mathbf{R}_j^{(\tau')})(\vec{\mathbf{x}}) \\
fol(\mathbf{R}_i^{(\tau)} \not\equiv_{ext} \mathbf{R}_j^{(\tau')}) & = & \neg fol(\mathbf{R}_i^{(\tau)} \equiv_{ext} \mathbf{R}_j^{(\tau')}) \\
fol(\mathbf{R}_i^{(\tau)} \sqsubseteq_{int} \mathbf{R}_j^{(\tau')}) & = & \forall \vec{\mathbf{x}}. \top_{[\tau]_i}(\vec{\mathbf{x}}) \wedge \top_{[\tau']_j}(\vec{\mathbf{x}}) \supset (fol(\mathbf{R}_i^{(\tau)})(\vec{\mathbf{x}}) \supset fol(\mathbf{R}_j^{(\tau')})(\vec{\mathbf{x}})) \\
fol(\mathbf{R}_i^{(\tau)} \not\sqsubseteq_{int} \mathbf{R}_j^{(\tau')}) & = & \neg fol(\mathbf{R}_i^{(\tau)} \sqsubseteq_{int} \mathbf{R}_j^{(\tau')}) \\
fol(\mathbf{R}_i^{(\tau)} \equiv_{int} \mathbf{R}_j^{(\tau')}) & = & \forall \vec{\mathbf{x}}. \top_{[\tau]_i}(\vec{\mathbf{x}}) \wedge \top_{[\tau']_j}(\vec{\mathbf{x}}) \supset (fol(\mathbf{R}_i^{(\tau)})(\vec{\mathbf{x}}) \equiv fol(\mathbf{R}_j^{(\tau')})(\vec{\mathbf{x}})) \\
fol(\mathbf{R}_i^{(\tau)} \not\equiv_{int} \mathbf{R}_j^{(\tau')}) & = & \neg fol(\mathbf{R}_i^{(\tau)} \equiv_{int} \mathbf{R}_j^{(\tau')})
\end{array}$$

- If C_i and C_j are two concepts over the alphabets Σ_i and Σ_j respectively, then:

$$\begin{array}{rcl}
fol(C_i \sqsubseteq_{ext} C_j) & = & \forall x. fol(C_i)(x) \supset fol(C_j)(x) \\
fol(C_i \not\sqsubseteq_{ext} C_j) & = & \neg fol(C_i \sqsubseteq_{ext} C_j) \\
fol(C_i \equiv_{ext} C_j) & = & \forall x. fol(C_i)(x) \equiv fol(C_j)(x) \\
fol(C_i \not\equiv_{ext} C_j) & = & \neg fol(C_i \equiv_{ext} C_j) \\
fol(C_i \sqsubseteq_{int} C_j) & = & \forall x. \top_{\mathcal{C}i}(x) \wedge \top_{\mathcal{C}j}(x) \supset (fol(C_i)(x) \supset fol(C_j)(x))
\end{array}$$

[2]We remark that the mapping focuses on the structural aspects of Object-Oriented Models, and does not consider their dynamic aspects.

$$
\begin{array}{rcl}
\mathit{fol}(C_i \not\sqsubseteq_{int} C_j) & = & \neg \mathit{fol}(C_i \sqsubseteq_{int} C_j) \\
\mathit{fol}(C_i \equiv_{int} C_j) & = & \forall x. \top_{\mathcal{C}i}(x) \wedge \top_{\mathcal{C}j}(x) \supset (\mathit{fol}(C_i)(x) \equiv \mathit{fol}(C_j)(x)) \\
\mathit{fol}(C_i \not\equiv_{int} C_j) & = & \neg \mathit{fol}(C_i \equiv_{int} C_j)
\end{array}
$$

Let L_i, L_j be relations or concepts defined, respectively, over the alphabets Σ_i and Σ_j. A first-order interpretation $\mathcal{I}$ over $\Sigma_i \cup \Sigma_j$ *satisfies* an intermodel assertion $L_i \propto L_j$ if $\mathit{fol}(L_i \propto L_j)$ evaluates to true in $\mathcal{I}$.

A *Conceptual Data Warehouse Model (CDWM)* $\mathcal{W}$ over the alphabet $\Sigma_{\mathcal{W}} = \Sigma_0 \dot\cup \Sigma_1 \dot\cup \ldots \dot\cup \Sigma_m$ and domains $\Sigma_{\mathcal{D}}$ is an $(m+2)$-tuple $\langle \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m, \mathcal{G} \rangle$ such that:

- $\mathcal{M}_0$ is the Enterprise Model, and each $\mathcal{M}_i$ is a Source Model (for each $i \in \{1, \ldots, m\}$).
- $\mathcal{M}_i$ is a $\mathcal{DLR}$ conceptual model (i.e. a finite set of intramodel assertions) over the alphabet Σ_i and domains $\Sigma_{\mathcal{D}}$, for each $i \in \{0, \ldots, m\}$. We assume that Σ_0 contains, for each atomic relation $\top_{[\tau]_i}$ appearing in some Σ_i ($i \in \{1, \ldots, m\}$), a corresponding atomic relation $\top_{[\tau]_0}$.
- $\mathcal{G}$ (for "glue") is a finite set of intermodel assertions, such that each element of $\mathcal{G}$ is an intermodel assertion over Σ_i, Σ_j ($i \neq j$), for some $i, j \in \{0, \ldots, m\}$. We assume that $\mathcal{G}$ always includes for each $i \in \{1, \ldots, m\}$ the following assertions:

$$
\begin{array}{lll}
\top_{\mathcal{C}i} \sqsubseteq_{ext} \top_{\mathcal{C}0} & & \\
\top_{[\tau]_i} \sqsubseteq_{ext} \top_{[\tau]_0} & & \text{for each } [\tau] \text{ in } \mathcal{M}_i
\end{array}
$$

In order to take into account concrete domains, we restrict again our interest to admissible interpretations of a CDWM $\mathcal{W}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over the alphabet $\Sigma_{\mathcal{W}}$ is *admissible* if the following conditions are satisfied:

1. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathcal{C}} \dot\cup \Delta_{\mathcal{D}}$;
2. $(\lambda x. D(x))^{\mathcal{I}} = \mathit{dom}(D) \qquad$ for each $D \in \Sigma_{\mathcal{D}}$;
3. $v^{\mathcal{I}} = \alpha(v) \qquad$ for each value of each $D \in \Sigma_{\mathcal{D}}$;
4. $\top_{\mathcal{C}0}^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathcal{C}}$;
5. $\top_{[\langle U_1:t_1, \ldots, U_n:t_n \rangle]_i}^{\mathcal{I}} \subseteq \Delta_1 \times \cdots \times \Delta_n, \qquad \Delta_i = \begin{cases} \top_{\mathcal{C}i}^{\mathcal{I}} & \text{if } t_i = \mathit{obj} \\ \Delta_{\mathcal{D}} & \text{if } t_i = \mathit{val} \end{cases} \qquad$ for each $\top_{[\tau]_i} \in \Sigma_i$.

Finally, an interpretation $\mathcal{I}$ for a CDWM $\mathcal{W}$ is a *model* for $\mathcal{W}$ if and only if the following conditions hold:

1. $\mathcal{I}$ is an admissible interpretation;
2. $\mathcal{I}$ satisfies all the intramodel assertions in $\mathcal{M}_i$ ($i \in \{0, \ldots, m\}$);
3. $\mathcal{I}$ satisfies all the intermodel assertions in $\mathcal{G}$.

Observe that, in order to extract from $\mathcal{I}$ an interpretation $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$ for each conceptual model $\mathcal{M}_i$ ($i \in \{), \ldots, m\}$), it suffices to define:

- $\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i}_{\mathcal{C}} \cup \Delta_{\mathcal{D}}$ with $\Delta^{\mathcal{I}_i}_{\mathcal{C}} = \top_{\mathcal{C}i}^{\mathcal{I}}$;
- $A_i^{\mathcal{I}_i} = A_i^{\mathcal{I}}$;
- $\mathbf{P}_i^{\mathcal{I}_i} = \mathbf{P}_i^{\mathcal{I}}$.

We illustrate the method by enriching the Enterprise and Source Models of the example of Section 4.3 with suitable intermodel assertions. The whole situation can be modeled by means of a CDWM $\mathcal{W} = \langle \mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{G} \rangle$, where the alphabets Σ_0, Σ_1 and Σ_2, and the signatures τ, τ', and τ'' are defined as in

Section 4.3. The intermodel assertions representing the various interdependencies between relations and concepts in different models are collected in $\mathcal{G}$, which is constituted by the following assertions:

$$\begin{array}{rcl}
\texttt{Department}_1 & \equiv_{ext} & \texttt{PrDept}_0 \\
\texttt{REG-AT}_1 & \sqsubseteq_{ext} & \texttt{REG-AT}_0 \\
\texttt{Client}_1 & \equiv_{ext} & \texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{REG-AT}_0[\texttt{cli}].\texttt{PrDept}_0 \\
\texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{CONTRACT}_0[\texttt{cli}].\top_{[\tau]} & \sqsubseteq_{ext} & \exists^{\geq 1}\texttt{PROMOTION}_1[\texttt{cli}].\top_{[\tau']} \\
\\
\texttt{Client}_2 & \sqsubseteq_{ext} & \texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{CONTRACT}_0[\texttt{cli}].\top_{[\tau]} \\
\texttt{Department}_2 & \sqsubseteq_{ext} & \texttt{Department}_0 \\
\texttt{Service}_2 & \equiv_{ext} & \texttt{Service}_0 \\
\\
\texttt{Client}_1 & \equiv_{int} & \texttt{Client}_2 \\
\texttt{Department}_1 & \equiv_{int} & \texttt{Department}_2
\end{array}$$

6 The Formalism for the Logical Level

In this section we present the formalism we use for expressing both the Source Schemas and the Data Warehouse Schema. Such schemas are intended to provide a structural description of the content of both the sources and the materialized views in the Data Warehouse. The design of the formalism was driven by the following requirements:

- The Source Schemas and the Data Warehouse Schema should be described in terms of a formalism positioned at the logical level in the database terminology.
- The connection between the formulation of the logical structure of the schemas and the conceptual level of the representation should be explicit.
- The formalism should be able to support reasoning over both the logical and the conceptual level of the representation.

In order to meet the above requirements, we propose to express the logical schema (of both the sources and the Data Warehouse) in terms of a set of relation schemas. Each relation schema describes either an element of the Source Schema or the logical definition of a materialized view in the Data Warehouse. The connection to the Conceptual Data Warehouse Model is established by defining each relation as a conjunctive query over the elements of the Conceptual Data Warehouse Model[3]. Furthermore, we designed suitable techniques for reasoning over conjunctive queries in the formalism (see Section 7).

A *conjunctive query* q for a conceptual model $\mathcal{M}$ (either the Enterprise Model or a Source Model) over $\Sigma_{\mathcal{M}}$ is an open formula of the form

$$\lambda\vec{\mathbf{x}}.\exists\vec{\mathbf{y}}.\alpha(\vec{\mathbf{x}},\vec{\mathbf{y}})$$

where $\alpha(\vec{\mathbf{x}},\vec{\mathbf{y}})$ is a conjunction of atomic formulae whose predicates are atomic concepts and relations built over $\Sigma_{\mathcal{M}}$, having $\vec{\mathbf{x}},\vec{\mathbf{y}}$ as arguments. The *arity* of the conjunctive query is equal to the number of variables of $\vec{\mathbf{x}}$. It is also common to use, instead of the notation above, the following one:

$$\{\vec{\mathbf{x}} \mid \exists\vec{\mathbf{y}}.\alpha(\vec{\mathbf{x}},\vec{\mathbf{y}})\}$$

We observe that, since assertions on both relations and concepts can be expressed in $\mathcal{M}$, every predicate used in the conjunctive query can actually denote any complex relation or concept expressible in $\mathcal{DLR}$. This distinguishes our approach with respect to [DLNS91, LR96], where the idea of expressing conjunctive queries over Description Logics is explored, but relations appearing in queries are not part of the knowledge base.

[3]In defining such queries the objects of the conceptual representation need to be coded into values of the logical representation. The problems related to this issue will be dealt with in WP4. For the moment we assume that this is automatically done by an implicit coding function.

We observe that the mechanism for describing the logical content of a source is similar to the one adopted in Information Manifold [LSK95, KLSS95, LRO96], where sources are described in terms of views over the World view. However, we note that our conceptual formalisms is more powerful than the corresponding one in Information Manifold. Moreover, we have defined suitable reasoning techniques over conjunctive queries, as we will see in Section 7. For a study of the expressive power of Description Logics, we refer the reader to [CPL97].

We conclude the section with an example of conjunctive query. We refer again to the ongoing example. Suppose that the form of actual data in Source 1 can be described in terms of a relational table $\mathtt{Table}_1$ with three columns, one for the client, one for the department which the client is registered at, and one for the location of the department. Such a table can be specified by means of the following conjunctive query:

$$\mathtt{Table}_1 \;=\; \{x, y, w \mid \mathtt{REG\text{-}AT}_1(x, y) \wedge \mathtt{LOCATION}_1(y, w)\}$$

7 Reasoning

One of the main advantages of providing a formalization of the CDWM in terms of the Description Logic $\mathcal{DLR}$ and of expressing the logical schemas as sets of (conjunctive) queries on the CDWM is that the reasoning services in $\mathcal{DLR}$ can support several forms of reasoning on the CDWM and the logical schemas. This section is devoted to a discussion on such reasoning services and the corresponding techniques.

7.1 Reasoning on the Conceptual Data Warehouse Model

The typical kinds of deductions needed at the conceptual level in order to support the designer in applying the integration methodology presented in Section 8 include verifying the following properties with respect to a CDWM $\mathcal{W}$:

- *Consistency of the CDWM*, i.e. whether $\mathcal{W}$ admits a model; a special case of this service concerns checking the consistency of the Enterprise Model or one Source Model in isolation.
- *Concept satisfiability*, i.e. whether $\mathcal{W}$ admits a model in which a certain concept has a nonempty extension.
- *Relation satisfiability*, i.e. whether $\mathcal{W}$ admits a model in which a certain relation has a nonempty extension.
- *Concept subsumption*, i.e. whether the extension of one concept is a subset of the extension of another concept in every model of $\mathcal{W}$.
- *Relation subsumption*, i.e. whether the extension of one relation is a subset of the extension of another relation in every model of $\mathcal{W}$.

Checking *concept satisfiability* can be considered as the basic reasoning task, since all reasoning tasks mentioned above can be reduced to it as follows:

- Checking whether $\mathcal{W}$ is consistent amounts to verify whether the concept $\top_{\mathcal{C}0}$ (which denotes the set of all objects in the interpretation domain) is satisfiable in $\mathcal{W}$. Checking whether a Source Model $\mathcal{M}_i$ is consistent in $\mathcal{W}$ amounts to verify whether the concept $\top_{\mathcal{C}i}$ is satisfiable in $\mathcal{W}$.
- Checking whether a relation $\mathbf{R}^{(\tau)}$ is satisfiable in $\mathcal{W}$ amounts to verify whether the concept $\exists^{\geq 1}\mathbf{R}^{(\tau)}[U].\mathbf{R}^{(\tau)}$ is satisfiable, where U is an attribute denoting a component of type *obj* of $\mathbf{R}^{(\tau)}$, i.e. $\tau = \langle \ldots, U\colon obj, \ldots\rangle$.
- Checking whether a concept C is subsumed by a concept C' in $\mathcal{W}$ amounts to verify whether $C \sqcap \neg C'$ is unsatisfiable in $\mathcal{W}$.
- Checking whether a relation $\mathbf{R}^{(\tau)}$ is subsumed by a relation $\mathbf{R}'^{(\tau')}$ in $\mathcal{W}$, where τ and τ' are compatible signatures, amounts to verify whether $\mathbf{R}^{(\tau)} \sqcap \neg\mathbf{R}'^{(\tau')}$ is unsatisfiable in $\mathcal{W}$.

These reasoning tasks can in particular be exploited for computing the concept and relation lattice of the CDWM, or more generally the lattice of all concept and relation expressions: this can be performed in an incremental manner by verifying subsumption between all pairs of concepts (relations) in the CDWM. Observe that such lattice can be maintained in an incremental manner (see Subsection 7.3).

Referring to the ongoing example, we can prove for instance that the assertion

$$\texttt{PROMOTION}_1 \sqsubseteq_{ext} \texttt{REG-AT}_0 \sqcap (\texttt{dep}\colon\texttt{PrDept}_0) \tag{1}$$

is a logical consequence of the CDWM (the reasoning service involved is relation subsumption). Indeed, observe that the CDWM includes the following assertions:

$$\begin{array}{rcll} \texttt{REG-AT}_1 & \sqsubseteq & (\texttt{cli}\colon\texttt{Client}_1) \sqcap (\texttt{dep}\colon\texttt{Department}_1) & (2)\\ \texttt{PROMOTION}_1 & \sqsubseteq & \texttt{REG-AT}_1 & (3)\\ \texttt{Department}_1 & \equiv_{ext} & \texttt{PrDept}_0 & (4)\\ \texttt{REG-AT}_1 & \sqsubseteq_{ext} & \texttt{REG-AT}_0 & (5) \end{array}$$

The intramodel assertions (2) and (3) imply that $\texttt{PROMOTION}_1 \sqsubseteq_{ext} (\texttt{dep}\colon\texttt{Department}_1)$, which, by the intermodel assertions (4) and (5), implies (1).

The expressiveness required by a data model to be able to express meaningful properties in the CDWM, and which is featured by $\mathcal{DLR}$, makes reasoning a complex task. We have devised a sound and complete procedure to decide the satisfiability of a concept with respect to a CDWM which works in worst-case deterministic exponential time in the size of the CDWM and the concept. Indeed, it is also possible to show that such worst-case complexity is inherent to the problem, proving that reasoning with respect to a CDWM is EXPTIME complete.

The inference method works in two steps: First, reasoning on the CDWM is reduced to reasoning on a knowledge base expressed in the Description Logic $\mathcal{CIQ}$ (introduced in [DGL96]). Then reasoning procedures for $\mathcal{CIQ}$ which are based on a correspondence with logics of programs are exploited to provide the reasoning services on the $\mathcal{CIQ}$ knowledge base.

We first describe the Description Logic $\mathcal{CIQ}$ and discuss the correspondence between $\mathcal{CIQ}$ and modal logics of programs, which is needed to establish EXPTIME decidability of reasoning in $\mathcal{CIQ}$. Then we show how a CDWM can be polynomially transformed into a $\mathcal{CIQ}$ knowledge base in such a way that the reasoning tasks on the CDWM are mapped to reasoning tasks on the $\mathcal{CIQ}$ knowledge base.

7.1.1 The Description Logic $\mathcal{CIQ}$

The Description Logic $\mathcal{CIQ}$ is an expressive description logic in which complex *concept* and *role* expressions (ranged over by C and R respectively) are built starting form a set of concept and role names (ranged over by A and P respectively). Concepts denote sets of objects, while roles denote *binary* relations over the interpretation domain. In specifying the syntax of $\mathcal{CIQ}$, we need to further distinguish between basic roles (ranged over by Q) and arbitrary complex roles, which are essentially regular expressions over basic roles:

$$\begin{array}{rcl} C & ::= & \top \mid A \mid \neg C \mid C \sqcap C' \mid \forall R.C \mid \exists^{\geq k} Q.C \\ Q & ::= & P \mid P^- \\ R & ::= & P \mid R^- \mid R \sqcup R' \mid R \circ R' \mid R^* \mid id(C) \end{array}$$

We also use $\bot$ as an abbreviation for $\neg\top$ and $C_1 \sqcup C_2$ as an abbreviation for $\neg(\neg C_1 \sqcap \neg C_2)$.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an *interpretation domain* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ mapping every atomic concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and every atomic role P to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is systematically extended to complex concepts and roles according to the semantics of the constructs specified as follows ($\sharp S$ denotes the cardinality of the set S, and $(R^{\mathcal{I}})^*$

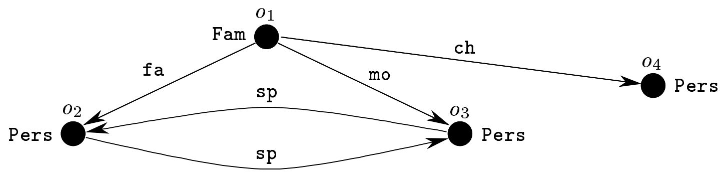

Figure 4: Instantiation of a schema — Transition system

stands for the reflexive transitive closure of the binary relation $R^{\mathcal{I}}$):

$$
\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{o \mid \forall o'.(o,o') \in R^{\mathcal{I}} \supset o' \in C^{\mathcal{I}}\} \\
(\exists^{\geq k} Q.C)^{\mathcal{I}} & = & \{o \mid \sharp\{o' \mid (o,o') \in Q^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \geq k\} \\
(R^-)^{\mathcal{I}} & = & \{(o,o') \mid (o',o) \in R^{\mathcal{I}}\} \\
(R_1 \sqcup R_2)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}} \\
(R_1 \circ R_2)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} \\
(R^*)^{\mathcal{I}} & = & (R^{\mathcal{I}})^* \\
(id(C))^{\mathcal{I}} & = & \{(o,o) \mid o \in C^{\mathcal{I}}\}
\end{array}
$$

A $\mathcal{CIQ}$ *knowledge base* is constituted by a finite set of *assertions* of the form

$$C_1 \sqsubseteq C_2 \qquad C_1 \not\sqsubseteq C_2 \qquad C_1 \equiv C_2 \qquad C_1 \not\equiv C_2.$$

An interpretation $\mathcal{I}$ *satisfies* an assertion $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, an assertion $C_1 \not\sqsubseteq C_2$ if $C_1^{\mathcal{I}} \not\subseteq C_2^{\mathcal{I}}$, an assertion $C_1 \equiv C_2$ if $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$, and an assertion $C_1 \not\equiv C_2$ if $C_1^{\mathcal{I}} \neq C_2^{\mathcal{I}}$. An interpretation which satisfies all assertions of a knowledge base $\mathcal{K}$ is a *model* of $\mathcal{K}$. The basic reasoning task in $\mathcal{CIQ}$ is checking *concept satisfiability* with respect to a knowledge base, which amounts to check whether the knowledge base admits a model in which a certain concept has a nonempty extension.

The main idea behind the inference method for $\mathcal{CIQ}$ is to take advantage of the strong similarity that exists between the interpretative structures of $\mathcal{CIQ}$ knowledge bases and transition systems used in computer science to describe the behavior of program schemes. This idea has already been exploited in previous work relating formalisms used in knowledge representation and databases to modal logics developed for modeling properties of programs [Sch91, CLN94, DGL94a, DGL95, DGL96, CDGL95].

To gain some intuition on the correspondence, consider Figure 4, showing an instantiation of a $\mathcal{CIQ}$ knowledge base, where nodes correspond to objects labeled by the concepts they belong to, and arcs correspond to roles. For example, o_1 is an instance of the concept `Fam` connected by means of roles `fa` and `mo` to two instances of the concept `Pers`. Now, such instantiation can also be seen as a transition system where nodes correspond to states labeled with the properties of the state, and arcs correspond to state transitions. For example, o_1 can be seen as a state where the property `Fam` holds, and such that the execution of program `fa` from it results in the state o_2, where `Pers` holds and `Fam` does not. Notice that the cycle involving o_2 and o_3 corresponds to a nonterminating computation.

The similarity between the interpretative structures in conceptual models and transition systems reflects in a tight correspondence, first pointed out in [Sch91], between Description Logics and a well-known class of modal logics of programs, Propositional Dynamic Logics (PDLs) [FL79, KT90, Sti92]. In particular, $\mathcal{CIQ}$ without qualified number restrictions corresponds to Converse PDL.

We make use of the correspondence between Description Logics and PDLs in order to exploit the sophisticated tools available for reasoning on logics of programs [Pra80, VW86, DGM96]. However, the high expressiveness of $\mathcal{CIQ}$, and in particular the presence of qualified number restrictions, requires to extend the reasoning techniques developed for Converse PDL. It turns out that the basic difficulty lies already in handling functionality, i.e. the simplest form of number restrictions. In [DGL94a] it is shown how reasoning on a knowledge base in the presence of functionality can be reduced to reasoning on a knowledge base without this constructor, and thus to reasoning in Converse PDL. The technique exploits reification of roles and is based on "pseudo axiomatization" of functionality by means of a finite set of assertions. The method has been extended in [DGL96] to deal also with the (qualified) number restrictions in $\mathcal{CIQ}$. Since the transformation used to eliminate both qualified number restrictions is polynomial in the size of the knowledge base[4], it follows from EXPTIME decidability of reasoning in Converse PDL [Pra79] that concept satisfiability in $\mathcal{CIQ}$ can be decided in worst case deterministic exponential time. In fact, from EXPTIME hardness of reasoning in PDL [FL79] it follows that concept satisfiability in $\mathcal{CIQ}$ is EXPTIME complete. Note that, although not present in [DGL96], assertions of the type $C_1 \not\sqsubseteq C_2$ and $C_1 \not\equiv C_2$ are straightforwardly dealt with by the transformation to Converse PDL.

7.1.2 Encoding of the Conceptual Data Warehouse Model in $\mathcal{CIQ}$

We show now how a CDWM $\mathcal{W}$ can be encoded in a $\mathcal{CIQ}$ knowledge base $\mathcal{K}_\mathcal{W}$ in such a way that checking the satisfiability of a $\mathcal{DLR}$ concept in $\mathcal{W}$ is reduced to checking the satisfiability of the corresponding $\mathcal{CIQ}$ concept in $\mathcal{K}_\mathcal{W}$. The encoding is based on a *reification* of n-ary relations, i.e. a transformation of atomic relations into atomic concepts such that each tuple of a relation is represented by an instance of the corresponding concept which is connected by means of functional roles (which correspond to the attributes of the relation) to the components of the tuple. The correctness of this technique is due to the inability in $\mathcal{CIQ}$ of expressing that two chains of links meet the same individual, thus disallowing the possibility of creating two individuals representing the same tuple [DGL94b, CLN94].

Let $\mathcal{W} = \{\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m, \mathcal{G}\}$ be a CDWM over $\Sigma_\mathcal{W} = \Sigma_0 \mathbin{\dot\cup} \Sigma_1 \mathbin{\dot\cup} \cdots \mathbin{\dot\cup} \Sigma_m$ with domains $\Sigma_\mathcal{D} = \{D_1, \ldots, D_h\}$. In the following we assume that relations, and concepts built over Σ_i are denoted by symbols subscripted with i. The knowledge base $\mathcal{K}_\mathcal{W}$ obtained from $\mathcal{W}$ contains an atomic concept for each atomic relation, atomic concept, domain, and domain value that appears in $\mathcal{W}$, and atomic roles $f_1, \ldots, f_{n_{max}}$, where n_{max} is the maximum arity of relations in $\mathcal{W}$. In order to specify the encoding of $\mathcal{W}$ into $\mathcal{K}_\mathcal{W}$ we introduce a mapping $\sigma(\cdot)$ as follows (we assume that τ and τ' are compatible signatures):

$$
\begin{array}{rcl}
\sigma(\mathbf{P}_i^{(\tau)}) & = & \top_{[\tau]_i} \sqcap \mathbf{P}_i^{(\tau)} \\
\sigma((U_\ell^{(\tau)}\!: C_i)) & = & \top_{[\tau]_i} \sqcap \exists f_\ell . \sigma(C_i) \\
\sigma((U_\ell^{(\tau)}\!: V)) & = & \top_{[\tau]_i} \sqcap \exists f_\ell . \sigma(V) \\
\sigma(\neg \mathbf{R}_i^{(\tau)}) & = & \top_{[\tau]_i} \sqcap \neg\sigma(\mathbf{R}_i^{(\tau)}) \\
\sigma(\mathbf{R}_i^{(\tau)} \sqcap \mathbf{R}_i'^{(\tau)}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \sqcap \sigma(\mathbf{R}_i^{(\tau)}) \\
\sigma(\delta_{\tau'}(\mathbf{R}_i^{(\tau)})) & = & \sigma(\mathbf{R}_i^{(\tau)}) \\
\sigma(A_i) & = & \top_{\mathcal{C}\,i} \sqcap A_i \\
\sigma(\neg C_i) & = & \top_{\mathcal{C}\,i} \sqcap \neg\sigma(C_i) \\
\sigma(C_i \sqcap C_i') & = & \sigma(C_i) \sqcap \sigma(C_i') \\
\sigma(\forall \mathbf{R}_i^{(\tau)}[U_\ell].\mathbf{R}_i'^{(\tau)}) & = & \top_{\mathcal{C}\,i} \sqcap \forall f_\ell^- . \neg\sigma(\mathbf{R}_i^{(\tau)}) \sqcup \sigma(\mathbf{R}_i'^{(\tau)}) \\
\sigma(\exists^{\geq k} \mathbf{R}_i^{(\tau)}[U_\ell].\mathbf{R}_i'^{(\tau)}) & = & \top_{\mathcal{C}\,i} \sqcap \exists^{\geq k} f_\ell^- . \sigma(\mathbf{R}_i^{(\tau)}) \sqcap \sigma(\mathbf{R}_i'^{(\tau)}) \\
\sigma(D) & = & D \\
\sigma(\{v_1, \ldots, v_h\}) & = & v_1 \sqcup \cdots \sqcup v_h \\
& & \\
\sigma(\mathbf{R}_i^{(\tau)} \sqsubseteq \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \sqsubseteq \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \not\sqsubseteq \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \not\sqsubseteq \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \equiv \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \equiv \sigma(\mathbf{R}_j'^{(\tau')})
\end{array}
$$

[4]As usual in Description Logics, we assume that numbers in qualified number restrictions are represented in unary.

$$
\begin{array}{rcl}
\sigma(\mathbf{R}_i^{(\tau)} \not\equiv \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \not\equiv \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \sqsubseteq_{ext} \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \sqsubseteq \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \not\sqsubseteq_{ext} \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \not\sqsubseteq \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \equiv_{ext} \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \equiv \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \not\equiv_{ext} \mathbf{R}_j'^{(\tau')}) & = & \sigma(\mathbf{R}_i^{(\tau)}) \not\equiv \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \sqsubseteq_{int} \mathbf{R}_j'^{(\tau')}) & = & \top_{[\tau]_i} \sqcap \top_{[\tau']_j} \sqsubseteq \neg\sigma(\mathbf{R}_i^{(\tau)}) \sqcup \sigma(\mathbf{R}_j'^{(\tau')}) \\
\sigma(\mathbf{R}_i^{(\tau)} \not\sqsubseteq_{int} \mathbf{R}_j'^{(\tau')}) & = & \top_{[\tau]_i} \sqcap \top_{[\tau']_j} \sqcap \sigma(\mathbf{R}_i^{(\tau)}) \sqcap \neg\sigma(\mathbf{R}_j'^{(\tau')}) \not\equiv \bot \\
\sigma(\mathbf{R}_i^{(\tau)} \equiv_{int} \mathbf{R}_j'^{(\tau')}) & = & \top_{[\tau]_i} \sqcap \top_{[\tau']_j} \sqsubseteq (\neg\sigma(\mathbf{R}_i^{(\tau)}) \sqcup \sigma(\mathbf{R}_j'^{(\tau')})) \sqcap (\sigma(\mathbf{R}_i^{(\tau)}) \sqcup \neg\sigma(\mathbf{R}_j'^{(\tau')})) \\
\sigma(\mathbf{R}_i^{(\tau)} \not\equiv_{int} \mathbf{R}_j'^{(\tau')}) & = & \top_{[\tau]_i} \sqcap \top_{[\tau']_j} \sqcap ((\sigma(\mathbf{R}_i^{(\tau)}) \sqcap \neg\sigma(\mathbf{R}_j'^{(\tau')})) \sqcup (\neg\sigma(\mathbf{R}_i^{(\tau)}) \sqcap \sigma(\mathbf{R}_j'^{(\tau')}))) \not\equiv \bot \\
\sigma(C_i \sqsubseteq C_j') & = & \sigma(C_i) \sqsubseteq \sigma(C_j') \\
\sigma(C_i \not\sqsubseteq C_j') & = & \sigma(C_i) \not\sqsubseteq \sigma(C_j') \\
\sigma(C_i \equiv C_j') & = & \sigma(C_i) \equiv \sigma(C_j') \\
\sigma(C_i \not\equiv C_j') & = & \sigma(C_i) \not\equiv \sigma(C_j') \\
\sigma(C_i \sqsubseteq_{ext} C_j') & = & \sigma(C_i) \sqsubseteq \sigma(C_j') \\
\sigma(C_i \not\sqsubseteq_{ext} C_j') & = & \sigma(C_i) \not\sqsubseteq \sigma(C_j') \\
\sigma(C_i \equiv_{ext} C_j') & = & \sigma(C_i) \equiv \sigma(C_j') \\
\sigma(C_i \not\equiv_{ext} C_j') & = & \sigma(C_i) \not\equiv \sigma(C_j') \\
\sigma(C_i \sqsubseteq_{int} C_j') & = & \top_{\mathcal{C}i} \sqcap \top_{\mathcal{C}j} \sqsubseteq \neg\sigma(C_i) \sqcup \sigma(C_j') \\
\sigma(C_i \not\sqsubseteq_{int} C_j') & = & \top_{\mathcal{C}i} \sqcap \top_{\mathcal{C}j} \sqcap \sigma(C_i) \sqcap \neg\sigma(C_j') \not\equiv \bot \\
\sigma(C_i \equiv_{int} C_j') & = & \top_{\mathcal{C}i} \sqcap \top_{\mathcal{C}j} \sqsubseteq (\neg\sigma(C_i) \sqcup \sigma(C_j')) \sqcap (\sigma(C_i) \sqcup \neg\sigma(C_j')) \\
\sigma(C_i \not\equiv_{int} C_j') & = & \top_{\mathcal{C}i} \sqcap \top_{\mathcal{C}j} \sqcap ((\sigma(C_i) \sqcap \neg\sigma(C_j')) \sqcup (\neg\sigma(C_i) \sqcap \sigma(C_j'))) \not\equiv \bot
\end{array}
$$

$\mathcal{K}_\mathcal{W}$ consists of the mappings $\sigma(\cdot)$ of all intra- and intermodel assertions in $\mathcal{W}$ and the following assertions:

$$
\begin{array}{rcl}
\top & \sqsubseteq & \top_\mathcal{R} \sqcup \top_{\mathcal{C}0} \sqcup \top_\mathcal{D} \\
\top_\mathcal{R} & \equiv & \bigsqcup_{\top_{[\tau]_0} \in \Sigma_0} \top_{[\tau]_0} \\
\top_\mathcal{R} & \sqsubseteq & \neg\top_{\mathcal{C}0} \sqcap \neg\top_\mathcal{D} \\
\top_{\mathcal{C}0} & \sqsubseteq & \neg\top_\mathcal{D} \\
\top_\mathcal{D} & \equiv & D_1 \sqcup \cdots \sqcup D_h \\
\top & \sqsubseteq & \neg\exists^{\geq 2} f_\ell.\top \qquad \text{for } \ell \in \{1,\ldots,n_{max}\} \\
\forall f_\ell.\bot & \sqsubseteq & \forall f_{\ell+1}.\bot \qquad \text{for } \ell \in \{1,\ldots,n_{max}-1\} \\
\top_{[\langle U_1:t_1,\ldots,U_n:t_n\rangle]_i} & \equiv & \exists f_1.\top_1 \sqcap \cdots \sqcap \exists f_n.\top_n \sqcap \forall f_{n+1}.\bot, \\
 & & \quad \text{with } \top_\ell = \begin{cases} \top_{\mathcal{C}i} & \text{if } t_\ell = obj \\ \top_\mathcal{D} & \text{if } t_\ell = val \end{cases} \quad \text{for } \ell \in \{1,\ldots,n\} \\
 & & \quad \text{for each } i \in \{0,\ldots,m\}, \text{ for each } \top_{[\langle U_1:t_1,\ldots,U_n:t_n\rangle]_i} \text{ in } \Sigma_i \\
\top_{\mathcal{C}0} \sqcup \top_\mathcal{D} & \sqsubseteq & \forall f_1.\bot \\
v & \sqsubseteq & D \qquad \text{for each value } v \text{ associated to a domain } D \text{ appearing in } \Sigma_\mathcal{W} \\
v & \sqsubseteq & \neg v' \qquad \text{for each pair of values } v \neq v' \text{ appearing in } \Sigma_\mathcal{W} \\
\mathbf{P}_i^{(\tau)} & \sqsubseteq & \top_{[\tau]_i} \qquad \text{for each } i \in \{0,\ldots,m\}, \text{ for each } \mathbf{P}_i^{(\tau)} \text{ in } \Sigma_i \\
A_i & \sqsubseteq & \top_{\mathcal{C}i} \qquad \text{for each } i \in \{0,\ldots,m\}, \text{ for each } A_i \text{ in } \Sigma_i
\end{array}
$$

The size of $\mathcal{K}_\mathcal{W}$ is polynomial in the size of $\mathcal{W}$.

The encoding of $\mathcal{DLR}$ relations and concepts in $\mathcal{CIQ}$ concepts and the additional assertions included in $\mathcal{K}_\mathcal{W}$ ensure that for every model $\mathcal{I}$ of $\mathcal{W}$ there is a corresponding model $\mathcal{I}_\mathcal{K}$ of $\mathcal{K}_\mathcal{W}$ such that for each (complex) relation and concept L in $\Sigma_\mathcal{W}$, every instance of L in $\mathcal{I}$ has a corresponding instance of $\sigma(L)$ in $\mathcal{I}_\mathcal{K}$. Conversely, it is possible to show that if $\mathcal{K}_\mathcal{W}$ admits a model $\mathcal{I}_\mathcal{K}$ then it also admits a model $\mathcal{I}'_\mathcal{K}$ which directly corresponds to a model of $\mathcal{W}$. In particular, the conditions that enforce $\mathcal{I}'_\mathcal{K}$ to correspond to an admissible interpretation are imposed explicitly in $\mathcal{K}_\mathcal{W}$ by suitable assertions. The concrete domains in $\mathcal{W}$ are represented in $\mathcal{K}_\mathcal{W}$ by atomic concepts which are enforced to be disjoint from the concepts representing $\mathcal{DLR}$ relations and concepts, while the domain value names are also represented by an atomic concepts which are forced to be mutually disjoint. The correctness of this approach is guaranteed by the use of signatures of relations in $\mathcal{DLR}$ and by the resulting syntactic restrictions, which essentially make it impossible to explicitly or implicitly enforce conditions on the domains in $\Sigma_\mathcal{D}$.

This allows us to prove the following theorem, and thus to reduce reasoning with respect to a CDWM to reasoning with respect to a $\mathcal{CIQ}$ knowledge base.

Theorem 1 *Let $\mathcal{W}$ be a CDWM over $\Sigma_\mathcal{W}$, $\mathcal{K}_\mathcal{W}$ be the corresponding $\mathcal{CIQ}$ knowledge base, and C be a concept over $\Sigma_\mathcal{W}$. Then C is satisfiable with respect to $\mathcal{W}$ if and only if $\sigma(C)$ is satisfiable with respect to $\mathcal{K}_\mathcal{W}$.*

7.1.3 Characterization of Reasoning with respect to a CDWM

By the EXPTIME upper bound on reasoning in $\mathcal{CIQ}$ and by applying Th. 1 we immediately obtain an EXPTIME upper bound for reasoning on a CDWM. On the other hand, by exploiting the same reduction used in [FL79] to show EXPTIME hardness of reasoning in PDL, we can easily obtain the same lower bound for reasoning on a CDWM. Thus we get a tight characterization of reasoning on a CDWM.

Theorem 2 *Checking concept satisfiability (and thus CDWM consistency, Source and Enterprise Model consistency, concept subsumption, relation satisfiability, and relation subsumption) with respect to a CDWM is an EXPTIME complete problem.*

7.2 Reasoning Taking into Account the Logical Level

In this subsection we discuss the reasoning techniques for the logical schema. More precisely, we show that reasoning in $\mathcal{CIQ}$ is once again the basic building block; however, reasoning on conjunctive queries requires a significant extension of the above described technique in order to cope with the variables of the query. Such an extension, which was first proposed in [CDGL97a], will be presented below.

Given two conjunctive queries q_1 and q_2 (of the same arity n) for a CDWM $\mathcal{W}$, we say that q_1 is *contained in* q_2 with respect to $\mathcal{W}$, if the set of tuples denoted by q_1 is contained in the set of tuples denoted by q_2 in every model of $\mathcal{W}$.

The decision problem we address is that of deciding whether a conjunctive query q_1 is contained in a conjunctive query q_2 with respect to $\mathcal{W}$. When restricting ourselves to considering only admissible interpretations, conjunctive query containment can be reformulated as logical implication as follows

$$fol(\mathcal{W}) \models_{adm} \forall \vec{\mathbf{x}}. q_1(\vec{\mathbf{x}}) \supset q_2(\vec{\mathbf{x}})$$

where the symbol $\models_{adm}$ denotes implication with respect to admissible interpretations, and $fol(\mathcal{W})$ is the conjunction of the first order translations of all intermodel and intramodel assertions in $\mathcal{W}$,

Referring again to the ongoing example consider the two queries posed on the Data Warehouse Model

$$\begin{aligned} q_1 &= \{x, y \mid \exists z. \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z)\} \\ q_2 &= \{x, y \mid \exists w, z. \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \wedge \mathtt{REG\text{-}AT}_0(x,w) \wedge \mathtt{PrDept}_0(w)\} \end{aligned}$$

It is immediate to see that q_2 is contained in q_1, even without taking the CDWM into account. However, also the converse containment can be proved. Indeed, from assertion (1), which we proved above, it follows that

$$\exists^{\geq 1}\mathtt{PROMOTION}_1[\mathtt{cli}].\top_{[\tau']} \sqsubseteq_{ext} \exists^{\geq 1}(\mathtt{REG\text{-}AT}_0 \sqcap (\mathtt{dep}\colon \mathtt{PrDept}_0))[\mathtt{cli}].\top_{[\tau']}.$$

This, together with the intermodel assertion

$$\texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{CONTRACT}_0[\texttt{cli}].\top_{[\tau]} \sqsubseteq_{ext} \exists^{\geq 1}\texttt{PROMOTION}_1[\texttt{cli}].\top_{[\tau']}$$

in the CDWM implies that q_1 is contained in q_2.

Let $q_1 = \lambda\vec{\mathbf{x}}.\exists\vec{\mathbf{y}}.\alpha(\vec{\mathbf{x}},\vec{\mathbf{y}})$ and $q_2 = \lambda\vec{\mathbf{x}}.\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{x}},\vec{\mathbf{z}})$ be two conjunctive queries for $\mathcal{W}$. Then q_1 is contained in q_2, i.e. $fol(\mathcal{W}) \models_{adm} \forall\vec{\mathbf{x}}.(\exists\vec{\mathbf{y}}.\alpha(\vec{\mathbf{x}},\vec{\mathbf{y}}) \supset \exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{x}},\vec{\mathbf{z}}))$, if and only if $fol(\mathcal{W}) \wedge \exists\vec{\mathbf{x}}.\exists\vec{\mathbf{y}}.\alpha(\vec{\mathbf{x}},\vec{\mathbf{y}}) \wedge \neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{x}},\vec{\mathbf{z}})$ is unsatisfiable in an admissible interpretation, if and only if

$$fol(\mathcal{W}) \wedge \alpha(\vec{\mathbf{a}},\vec{\mathbf{b}}) \wedge \neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}}) \tag{6}$$

is unsatisfiable in an admissible interpretation, where $\vec{\mathbf{a}}$, $\vec{\mathbf{b}}$ are distinct constants.

We develop an algorithm for deciding conjunctive query containment with respect to a CDWM which exploits a reduction to unsatisfiability in $\mathcal{CIQ}$. The algorithm extends the one developed in [CDGL97a] to decide conjunctive query containment with respect to $\mathcal{DLR}$ knowledge bases in order to deal with intra- and intermodel assertions. The technique is based on an encoding of the three parts of formula (6) in a $\mathcal{CIQ}$ knowledge base $\mathcal{K} = \mathcal{K}_\mathcal{W} \cup \mathcal{K}_\alpha \cup \mathcal{K}_\beta \cup \mathcal{K}_{aux}$ and a concept C_{root}, and on reducing containment of q_1 in q_2 with respect to $\mathcal{W}$ to unsatisfiability of C_{root} with respect to $\mathcal{K}$. We describe now how the various parts of $\mathcal{K}$ and C_{root} are constructed.

Encoding of $\mathcal{W}$

$\mathcal{K}_\mathcal{W}$ is the encoding of $\mathcal{W}$ into a $\mathcal{CIQ}$ knowledge base, as defined in Section 7.1.2.

Encoding of $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$

$\mathcal{K}_\alpha$ encodes $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$, which is of the form

$$\mathbf{P}_1(\vec{\mathbf{c}}_{\mathbf{P}_1}) \wedge \cdots \wedge \mathbf{P}_p(\vec{\mathbf{c}}_{\mathbf{P}_p}) \wedge A_1(c_{A_1}) \wedge \cdots \wedge A_a(c_{A_a}).$$

In $\mathcal{K}_\alpha$ we make use of special concepts, called *sk-concepts*, for representing the constants in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$ (the properties of sk-concepts are specified by $\mathcal{K}_{aux}$ – see later). More specifically, we introduce one sk-concept c for each constant c in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$, and one sk-concept $\vec{\mathbf{c}}$ for each $\mathbf{P}(\vec{\mathbf{c}})$ in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$.

For each $\mathbf{P}(\vec{\mathbf{c}})$ in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$ we include in $\mathcal{K}_\alpha$

$$\vec{\mathbf{c}} \sqsubseteq \mathbf{P}$$

and for each $A(c)$ in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$ we include

$$c \sqsubseteq A.$$

For each $\vec{\mathbf{c}} = c_1,\ldots,c_n$ we include

$$\vec{\mathbf{c}} \equiv \exists f_1.c_1 \sqcap \cdots \sqcap \exists f_n.c_n \sqcap \forall f_{n+1}.\bot$$

and for each c_i, $i \in \{1,\ldots,n\}$, we include

$$c_i \sqsubseteq \neg\exists^{\geq 2} f_i^-.\vec{\mathbf{c}}.$$

The size of $\mathcal{K}_\alpha$ is polynomial in the size of q_1.

Intuitively, $\mathcal{K}_\alpha$ is very close to an encoding of a set of assertions on objects regarding membership in concepts and participation in relations (such a set of assertions on objects is called an *ABox*) as a set of assertions on concepts corresponding to the objects. In principle, we build an ABox corresponding to $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$ and then we encode it as assertions on concepts introducing the sk-concepts representing objects. The encoding is that of [DGL96], except that: (i) we allow two objects in the ABox to denote the same element of the interpretation domain; (ii) we force objects representing tuples to be completely determined by the objects representing their components.

Encoding of $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$

We call *parameters* all a_i and z_j appearing in $\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$. $\mathcal{K}_\beta$ encodes $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$ by making use of the so called *tuple-graph*, which is a directed graph having:

- One node for each parameter v and one node for each $\vec{\mathbf{v}}$ such that $\mathbf{P}(\vec{\mathbf{v}})$ appears in $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$. Each node corresponding to v is labeled by v itself and all A such that $A(v)$ appears in $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$. Similarly, each node corresponding to $\vec{\mathbf{v}}$ is labeled by $\vec{\mathbf{v}}$ and all $\mathbf{P}$ such that $\mathbf{P}(\vec{\mathbf{v}})$ appears in $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$.
- For each node labeled by $\vec{\mathbf{v}} = v_1,\ldots,v_n$, one edge labeled by i to the node labeled by v_i, for $i \in \{1,\ldots,n\}$.

In general the tuple-graph may be composed of $m \geq 1$ connected components. For the i-th connected component we build a concept expression $\delta_i(\vec{\mathbf{z}})$ by starting from a node corresponding to a parameter and visiting the corresponding component as follows (let u be the current node in the visit):

- If u corresponds to a parameter v, then construct the concept as the conjunction of: (i) v, if u is either a_i, or a z_j appearing in a cycle in the tuple-graph, $\top_1$ otherwise; (ii) every concept labeling the node u; (iii) one concept $\exists f_i^-.C$ for each non-marked edge labeled by i from a node $\vec{\mathbf{v}}$ to v, where C is the concept resulting by marking the edge and visiting the node corresponding to $\vec{\mathbf{v}}$.
- If u corresponds to a tuple $\vec{\mathbf{v}} = v_1,\ldots,v_n$, let $e_1,\ldots,e_h$ be the non-marked edges from u to the nodes corresponding to the components v_i. We mark $e_1,\ldots,e_h$ and construct the concept as the conjunction of: (i) every relation labeling the node u; (ii) one concept $\exists f_i.C$ for each edge e_j, where i is the label of e_j, and C is the concept resulting by visiting the node corresponding to v_i.
- If u has already been visited then it must correspond to a parameter v and the resulting concept is v.

Observe, that $\delta_i(\vec{\mathbf{z}})$ contains newly introduced atomic concepts z_j, one for each variable occurring in a cycle in the tuple-graph.

$\mathcal{K}_\beta$ consists of all assertions of the form ($\mathbf{u}$ abbreviates $(f_1 \sqcup \cdots \sqcup f_{n_{max}} \sqcup f_1^- \sqcup \cdots \sqcup f_{n_{max}}^-)^*$)

$$\top \ \sqsubseteq\ \neg(\exists\mathbf{u}.\delta_1(\vec{\mathbf{z}}) \sqcap \cdots \sqcap \exists\mathbf{u}.\delta_m(\vec{\mathbf{z}}))$$

obtained by replacing each z_j occurring in the assertion above by each sk-concept corresponding to a constant in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$. Observe that the number of assertions in $\mathcal{K}_\beta$ is $O(\ell_1^{\ell_2})$, where ℓ_1 is the number of variables in q_1 and ℓ_2 is the number of variables z_j occurring in a cycle in the tuple-graph.

The special attention that $\mathcal{K}_\beta$ needs is due to the fact that the variables $\vec{\mathbf{z}}$ in $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$ are universally quantified. Conceptually we need to distinguish between two cases, depending on whether there is a cycle in the *tuple-graph* for q_2. The tuple-graph for a query reflects the dependency between variables and tuples resulting from the appearance of the variables in the atoms:

- If there is no cycle in the tuple-graph then we can directly build a concept expressing the constraints in $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$.
- If there is a cycle then, due to the fundamental inability of expressing in Description Logics that two chains of links meet the same individual, no concept can directly express the constraints in $\neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}},\vec{\mathbf{z}})$. For the same reason, however, the only cycles that can be enforced in the models are those formed by the objects corresponding to the constants $\vec{\mathbf{a}},\vec{\mathbf{b}}$. Therefore it suffices to build a concept for each possible instantiation for the variables z_i, appearing in some cycle of the tuple-graph, with the objects corresponding to $\vec{\mathbf{a}},\vec{\mathbf{b}}$.

Encoding of objects

$\mathcal{K}_{aux}$ enforces that for each sk-concept representing a constant in $\alpha(\vec{\mathbf{a}},\vec{\mathbf{b}})$, a specific instance representative of the constant can be singled out. This is done, similarly to [DGL96] by one assertion of the form

$$\begin{array}{rcl}(\vec{\mathbf{c}} \sqcap C) & \sqsubseteq & \forall\mathbf{u}.(\neg\vec{\mathbf{c}} \sqcup C)\\ (c \sqcap C) & \sqsubseteq & \forall\mathbf{u}.(\neg c \sqcup C)\end{array}$$

for each sk-concept $\vec{\mathbf{c}}$, c in $\mathcal{K}_\alpha$, and each concept C satisfying the conditions specified in [DGL96]. The number of such assertions is polynomial in the size of $\mathcal{K}_\mathcal{W} \cup \mathcal{K}_\alpha \cup \mathcal{K}_\beta$.

The concept C_{root}

Let $s_1, \ldots, s_t$ be all sk-concepts. Then

$$C_{root} \;=\; \exists create.s_1 \sqcap \cdots \sqcap \exists create.s_t$$

where *create* is a newly introduced atomic role. Thus C_{root} expresses the existence of an instance for each of the sk-concepts.

Theorem 3 *$fol(\mathcal{W}) \wedge \alpha(\vec{\mathbf{a}}, \vec{\mathbf{b}}) \wedge \neg\exists\vec{\mathbf{z}}.\beta(\vec{\mathbf{a}}, \vec{\mathbf{z}})$ is unsatisfiable in an admissible interpretation if and only if C_{root} is unsatisfiable in $\mathcal{W}$.*

As already mentioned, satisfiability of a concept in a $\mathcal{CIQ}$ knowledge base is an EXPTIME complete problem, and thus we get the following upper bound for conjunctive query containment.

Theorem 4 *Deciding whether a conjunctive query q_1 is contained in a conjunctive query q_2 with respect to a CDWM $\mathcal{W}$ can be done in time $O(2^{p(|\mathcal{W}|\cdot \ell_1^{\ell_2})})$, where $|\mathcal{W}|$ is the size of $\mathcal{W}$, ℓ_1 is the number of variables in q_1, and ℓ_2 is the number of existentially quantified variables in q_2 that appear in a cycle of the tuple-graph for q_2.*

7.3 On the Complexity of Reasoning

In this subsection we discuss the complexity of reasoning, with the goal of identifying guidelines for the implementation and the evaluation of reasoning systems providing the above described inference services. Such guidelines will be refined and detailed in the future activities of WP3.

The first observation is that the above complexity results, showing that in the worst-case the problems are computationally very hard, are being balanced by practical studies [Hor97] showing that reasoning tasks encountered in practice can, in the average, be solved with acceptable performance. Moreover, there are several proposals for efficient implementation and effective optimization of reasoning on conceptual data models (see for example [CL94a]), that can be applied in our framework.

A second aspect to be noticed is that if a greater effort at design time leads to more effective operational systems, the computational cost of the reasoning processes can have a very good pay-off. This also suggests that the problem can be attacked by pre-computing suitable structures that can allow for better efficiency at query time.

In fact, we propose to embed the knowledge represented in a CDWM $\mathcal{W}$, and in particular, by a set of intermodel assertions, into a compact structure, reflecting all the relevant interdependencies holding between relations and concepts of different schemas. Specifically, in our approach, the relevant intermodel knowledge can be expressed in terms of a so-called *intermodel knowledge network*, which is defined as a directed graph whose nodes represent relations or concepts, and arcs represent extensional containment between them.

The construction of the intermodel knowledge network $N^\mathcal{W}$ associated with a CDWM $\mathcal{W}$ is based on the reasoning technique described in Subsection 7.1, and proceeds as follows:

1. select from the models $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$ the relations and concepts that are to be associated with the nodes of $N^\mathcal{W}$,

2. for every pair of selected relations or concepts L_1, L_2, draw an arc in $N^\mathcal{W}$ from the node corresponding to L_1 to the node corresponding to L_2, if $L_1 \equiv_{ext} L_2$ follows form $\mathcal{W}$

3. compute the transitive reduction of $N^\mathcal{W}$, i.e. iteratively remove from $N^\mathcal{W}$ all the arcs from node n_1 to node n_2 such that there is a node n_3 with arcs (n_1, n_3) and (n_3, n_2) in $N^\mathcal{W}$,

4. for every cycle γ in $N^\mathcal{W}$, merge all the nodes belonging to γ into a single node.

By comparing the intermodel knowledge network $N^{\mathcal{W}}$ with $\mathcal{W}$ itself, we can point out three main characteristics of $N^{\mathcal{W}}$.

First, $N^{\mathcal{W}}$ refers to a selected subset of (not necessarily all) the relations and concepts in the various schemas constituting $\mathcal{W}$. There are several possibilities for such a selection. For example, we may select only atomic relations and concepts from the various models, or we may include all atomic elements plus some important complex ones, or we may want to represent in $N^{\mathcal{W}}$ all the relations and concepts denoted by some expression in the intermodel assertions. Note that the choice of which entities are represented in $N^{\mathcal{W}}$ determines both the complexity and the completeness of the information carried by $N^{\mathcal{W}}$ itself.

Second, while $\mathcal{W}$ is a flat collection of intermodel assertions, $N^{\mathcal{W}}$ reflects the structure of interdependencies between relations and concepts, by making explicit all the extensional inclusions between selected relations and concepts that are implied by $\mathcal{W}$. Note that such a structure results in a partial order between the selected relations and concepts.

Finally, the arcs of $N^{\mathcal{W}}$ reflect some important inferences that can be drawn from $\mathcal{W}$. In particular, the existence of a path in $N^{\mathcal{W}}$ from two nodes n_1 and n_2 implies that all instances of the relation or concept represented by n_1 is also an instance of the relation or concept represented by n_2. Moreover, the mutual extensional equivalence of a set of selected relations and concepts is made explicit in $N^{\mathcal{W}}$ by merging the corresponding nodes.

8 Methodology

In this section, we outline the methodology for source integration in DWQ, based on the techniques described in the previous sections. We point out that the refinement of the methodology is the main subject of the remaining portion of WP3.

The methodology identifies the following situations that need to be dealt with in a systematic manner in order to build and maintain the Conceptual Data Warehouse Model, the Source Schemas and the Data Warehouse Schema:

1. A new source (or, a new portion of a source) is analyzed and added to the whole architecture; the data integration design scenario corresponding to this situation will be called *source-driven*.

2. A new query is formulated by a client; the design scenario corresponding to this situation will be called *client-driven*.

Note that, although the two scenarios refer to a dynamic situation where a new piece of information or a new request is considered, one can in practice iterate the process to design the Data Warehouse from scratch.

In the following subsections, we describe the most important methodological aspects that are to be considered in the two scenarios.

8.1 Source-Driven Integration

Source-driven integration is triggered when a new source or a new portion of a source is taken into account for integration into the Data Warehouse. The steps to be accomplished in this case are the following:

1. *Source Model construction.* The Source Model corresponding to the new source is produced, if not available.

2. *Source Model integration.* The Source Model is *integrated into the Conceptual Data Warehouse Model*. This can lead to changes both to the Source Models, and to the Enterprise Model. Moreover, intermodel assertions between the Enterprise Model and the Source Models and between the new source and the existing sources are added to the Conceptual Data Warehouse Model.

3. *Quality analysis.* The Quality Factors of the resulting Conceptual Data Warehouse Model are evaluated and a restructuring of the Models is accomplished to match the required criteria. This step requires also the use of the reasoning techniques associated to our formalisms.

4. *Source Schema construction.* The Source Schema corresponding to the new source (or, corresponding to a new portion of the source) is produced.

5. *Data Warehouse Schema restructuring.* On the basis of the new analyzed source, an analysis is carried out on whether the Data Warehouse Schema should be restructured and/or modified.

6. *DWQ Repository update.* All the changes, the relevant inferences carried out in the preceding steps, and the design decisions taken during source integration are represented in the DWQ Repository.

In the following we address in more detail the above steps.

1. Source Model construction

The goal of this step is to produce the conceptual model of the source under analysis. Since in a typical Data Warehouse setting, sources already exist, this task may be accomplished through a reverse engineering activity [BCN92].

It is worth stressing that the Source Model is really meant to capture the concepts and the relationships that are critical for the enterprise, independently of the organization of the data recorded in the physical structures. To this end, our approach provides a very expressive modeling language, which, as already pointed out, embodies the features of the most popular data models. In addition, in our formalism, it is possible to reason about the Source Model. Once a formalization of the model in terms of $\mathcal{DLR}$ is provided, checking several interesting properties of the model becomes possible, and can be used to help correctness and optimality of the design. We refer to the final section of [CLN97] for a discussion on using the inference techniques associated to Description Logics during Source Model construction.

2. Source Model integration

The goal of this step is to integrate the Source Model constructed in Step 1 with the Enterprise Model. We remind the reader that our approach to integration is mainly declarative: the most important efforts during source integration are indeed devoted to single out and to specify the intermodel assertions relating the Enterprise Model and the Source Models, rather than producing a unified conceptual representation. More precisely, the step of Source Model integration is characterized by the following activities:

- Structural and semantic conflicts involving the Source Model under analysis are detected and solved.

- Intermodel assertions between the Source Model and the Enterprise Model, and between different sources, are added to the Conceptual Data Warehouse Model.

The activity of conflict resolution takes advantage of the large body of work in database integration, as surveyed in Section 2. More precisely, in our framework, the basic structural and semantic conflicts are very similar to those arising in the Entity-Relationship Data Model. An example of structural conflict is represented by the situation where the same concept is represented as a class in one model and as a relation in another model. The principles for resolving such conflicts are now well established [BCN92]. Other types of conflicts are dealt with in the Quality Analysis step.

The specification of intermodel assertions represents the novel part of the methodology. As we said in Section 5, there are two types of intermodel assertions, namely, assertions relating concepts in one Source Model with concepts in the Enterprise Model, and assertions relating concepts in different Source Models. Although the vast majority of assertions will be of the first type, we stress the importance of specifying inter-relationships between sources, when available. For example, specifying that the set of instances of concept C_i of source S_i is always a subset of those of C_j in source S_j can be important in order to infer that accessing source S_j for retrieving instances of C is useless. We point out that the possibility of expressing relationships between concepts in different sources is a distinguished feature of our approach. As a general observation, intermodel assertions can be roughly classified as follows:

- *Subsetting assertions*, that are used to state that a certain concept or relation in the Source Model is a subset of another concept or relation in the Enterprise Model (or in another source). These assertion have the form $L_i \sqsubseteq_{ext} L'_j$.

- *Definition assertions*, that are used to completely characterize the set of instances of one concept in a model in terms of the set of instances of a concept in another model.
- *Completeness assertions*, that are used to state that the set of instances of a concept or relation in the Enterprise Model can be obtained as the union of different concepts or relations in the various sources. A special case of this type of assertions is the one stating that a certain concept in the Enterprise Model is fully captured by a concept in one Source Model. This assertion formalizes the notion of "proprietary entity" used e.g. in the method adopted in Telecom Italia for source integration in their Data Warehouse project IBDA.
- *Synonym assertions*, that are used to state that different symbols in two models denote in fact the same concept. These assertions have the form $L_i \equiv_{int} L_j$.
- *Homonym assertions*, that are used to state that the same symbol is used to denote different concepts in different models. These assertions have the form $L_i \not\equiv_{int} L_j$.

It is important to observe that the possibility of using complex concept and relation expressions in the context of intermodel assertions greatly enhances the expressive power of such assertions, and is another distinguished feature of our approach. Moreover, the possibility of reasoning about intermodel assertions provides support and guidelines to the designer of the Conceptual Data Warehouse Model, as pointed out in the discussion on the step of quality analysis. Finally, we note that the usage of intermodel assertions to reason about queries will be addressed in the next subsection on client-driven integration.

3. Quality analysis

The goal of this step is to verify that the quality requirements are met by the Conceptual Data Warehouse Model. In particular, the reasoning capabilities of our approach allow for dealing with several quality factors, such as:

- *Consistency* of the Source Model in isolation.
- *Redundancy*, by identifying equivalent concepts.
- *Readability*, by pointing out relationships that are implicit in the model.
- *Accessibility*, which amounts to verifying which data are available in the Data Warehouse, which data can be extracted from the sources, and which are indeed needed from external sources.
- *Believability*, which amounts to verifying whether the data available in the warehouse or provided by a source are consistent and complete.

It is worth noticing that, depending on the result of this step, a restructuring of both the Source Model and the Enterprise Model may be needed.

4. Source Schema construction

The goal of this step is to produce or enrich the Source Schema, i.e. the logical view of the source expressed as a collection of conjunctive queries over the corresponding Source Model. As we said above, this step has the goal of characterizing the actual data in the source in terms of a logical model of data. This feature, together with the ability to reason about queries constitutes the basis for the analysis of the queries that will be addressed in the next subsection.

Notice that, in order to guarantee the access to the actual data of the source, a specific wrapper for the source has to be provided if not available. This aspect will be covered in the context of WP4 – Data Reconciliation.

5. Data Warehouse Schema restructuring

This step amounts to restructuring the Data Warehouse Schema, i.e. the logical view of the data that are actually materialized in the Data Warehouse. Also in this case, the view is constituted by a set of conjunctive queries over the Conceptual Data Warehouse Model. This step is also included in the client-driven sequence of actions to be addressed in the next subsection. In this case, the restructuring may be needed on the basis of new data available after the analysis of the new source (or the new portion of the source). Indeed, the availability of new concepts and relations in the Conceptual Data Warehouse Model, and the possibility of characterizing how such new concepts are stored in the sources, may result in a new organization of the materialized data in the Data Warehouse, in order to better meet quality requirements. For example, the designer may want to reconsider design questions such as

1. What is appropriate to materialize in the Data Warehouse?
2. What kind of aggregations are needed in the Data Warehouse?
3. How should the logical schema be organized in order to effectively represent the multidimensional modeling requirements?

Finally, notice that a restructuring of the Data Warehouse Schema may require the design of new mediators. This issue will be dealt with in the context of WP4 – Data Reconciliation.

6. DWQ Repository update

The goal of this step is to store the results of the above steps within the DWQ repository. As specified in Deliverable 2.1, the representation of the information in the repository is done using the TELOS language.

The basic structure of the integration knowledge base is explicitly represented in the repository, so that one can, for example, retrieve interesting information, such as a concept definition, or a set of related intermodel assertions. Moreover, the result of all subsumption checks carried out by the system during the above mentioned step, are stored in the integration knowledge base. The importance of storing such information has been stressed at the end of Section 7.

In particular, the links `isSubsumedBy` and `isNotSubsumedBy` of the architectural model described in the Deliverable 2.1 are maintained as follows.

- Whenever a new subsumption relation between C_i and C_j is computed, the pair C_1, C_2 is stored in `isSubsumedBy`.
- Whenever a subsumption check between C_i and C_j fails, the pair C_1, C_2 is stored in `isNotSubsumedBy`.
- Whenever a new assertion is stored in the Conceptual Data Warehouse Model, all links of type `isNotSubsumedBy` that can be affected by the new subsumption relation are canceled.
- Whenever an assertion is removed from the Conceptual Data Warehouse Model, all links of type `isSubsumedBy` that were dependent from the removed assertion are canceled.

In the remaining part of WP3 we will characterize in more detail the techniques we use to compute the subsumption relations that are affected by the removal of an assertion, and the non-subsumption relations that are affected by a new assertion.

8.2 Client-Driven Integration

The client-driven design scenario refers to the case when a new query (or a set of queries) posed by a client is considered. In this case, an activity of query analysis is carried out, and three different situations may arise, depending on the result of such analysis:

1. The answer can be computed given the set of materialized views defined in the Data Warehouse Schema.
2. The answer requires new data available in the sources, but not in the Data Warehouse.

3. The answer requires data that are available neither at the Data Warehouse nor at the analyzed sources.

Notice that each of the above scenarios should be regarded as an extension of the previous one. In fact, in the most complex case of new sources required, it may be necessary both to get part of the data from the existing sources and to get part of the data from the materialized views.

The analysis of the query can be done by systematically decomposing it and checking whether its components are subsumed by the Data Warehouse Schema and by the sources. Therefore, the central reasoning service for query analysis is query containment checking. The techniques for query analysis will be further developed in the remaining part of WP3.

1. Fully materialized data

In this case, the answer to the query can be computed on the materialized set of views of the Data Warehouse. Based on the definition of the Data Warehouse Schema, the designer can find optimal ways to compute the answer. However, this may require a reorganization of the Data Warehouse Schema, which is being addressed in WP8.

2. New data required from existing sources

In the case when a portion of the required data is not materialized in the Data Warehouse, but is accessible from the sources, two steps must be accomplished:

1. Find the "optimal" set of subqueries to compute the answer from the sources.
2. Extend and/or restructure the logical Data Warehouse Schema.

While the second step is again being addressed in WP8, we shall focus on the first one, which will be fully characterized in the final part of WP3. Below we sketch the basic idea.

Different optimal choices can be characterized, based on various preference criteria. In [LSK95], the minimization in terms of the number of sources is proposed, based on the observation that accessing a source is the most expensive part of the process. In fact, by exploiting the information available through the intermodel assertions, we can accommodate different kinds of constraints, that are related to the above mentioned quality factors. For example, we can optimize with respect to believability, or interpretability, possibly combining different factors.

3. New data from new sources

In this case, the data resulting from the analyzed sources are not sufficient to answer the query. Thus, the new data should be searched for in new sources, or in new portions of already analyzed sources. The new source (or the new portion of a source) is then added to the Conceptual Data Warehouse Model, and therefore we can resort to the source-driven sequence of steps, which has been addressed in the previous subsection.

The result of the integration of the new source can thus lead to a restructuring of the Conceptual Data Warehouse Model, concerning the specific Source Model, possibly the Enterprise Model and the intermodel assertions. In addition, the logical schema of the source may need to be created or updated. Finally, the Data Warehouse Schema, i.e. the logical view of the materialized data in the Data Warehouse, may be upgraded.

9 Conclusions

The results reported in the present document show that the tasks for the WP3 on Source Integration are being accomplished according to the workplan and significant progress has been made towards the WP objectives. In particular, we have achieved the following goals:

1. We have defined the architecture for source integration, which is based on a well-established structure with both a conceptual and a logical level, but embodies several novel features as compared with conventional database design.

2. We have characterized the modeling language by extending previously known formalisms for data modeling in several respects: expressiveness of the modeling language, introduction of multi-language structure to deal with the multi-source scenario of Data Warehouse applications, and intermodel assertions to precisely characterize the relationships between the Enterprise Model and the Source Models and between the Source Models.

3. We have proposed reasoning techniques that provides for the basic reasoning services both at the conceptual and at the logical level, by exploiting previous results on reasoning in expressive Description Logics and extending them to the case of conjunctive queries.

4. We have outlined a methodology for systematically adding, characterizing, and integrating information sources. The methodology supports two main design scenarios, called source-driven and client-driven, respectively. We have also outlined a method for maintaining the so-called integration knowledge base, stored in the DWQ Repository.

The remaining part of WP3 will be devoted to a further development both of the formalism for conceptual and logical modeling in order to take into account aggregated data, and of the methodology for source integration. It will also be investigated in more detail how unification and matching of concepts in description logics [BN97] can be used to support the source model integration (e.g., by suggesting synonyms). Moreover, we shall undertake the design of a prototype, based on an implemented Knowledge Representation System for Description Logics, which can provide active support to the user in the design process, as outlined in the present document. We will also study in detail the interaction between such a tool and the DWQ Repository.

More specifically, we shall address the problems of query analysis, answering queries using materialized views, and source optimization. In these tasks, we intend to focus the methodology on the quality factors that are to be taken into account at the design stage in order to fulfill the requirements for a given application.

The work carried out within WP3 is also the basis for the development of WP4 on Data Reconciliation, which follows, in the workplan, WP3. In WP4, based on the setting presented here, we can effectively address the issue of Wrapper and Mediator generation as well as the overall issue of data reconciliation. To this end, the framework we have defined provides us with all the necessary information both on the source structure and on the data displacement. In addition, by relying on a fairly expressive formalisms, where reasoning is still decidable, we have the right methodological tools to address the issue of object matching.

References

[Abr74] J. R. Abrial. Data semantics. In J. W. Klimbie and K. L. Koffeman, editors, *Data Base Management*, pages 1–59. North-Holland Publ. Co., Amsterdam, 1974.

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

[AKC96] Y. Arens, C. A. Knoblock, and W. Chen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6:99–130, 1996.

[Alb96] Joseph Albert. Data integration in the RODIN multidatabase system. In *Proc. of the 4rd Int. Conf. on Cooperative Information Systems (CoopIS-96)*, pages 48–57, 1996.

[BCN92] Carlo Batini, Stefano Ceri, and Sham B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach*. Benjamin and Cummings Publ. Co., Menlo Park, California, 1992.

[BCW90] Mokrane Bouzeghoub and Isabelle Comyn-Wattiau. View integration by semantic unification and transformation of data structures. In *Proc. of the 9th Int. Conf. on the Entity-Relationship Approach (ER-90)*, pages 413–430, 1990.

[BDK92] P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *Proc. of the 3rd Int. Conf. on Extending Database Technology (EDBT-92)*, Lecture Notes in Computer Science, pages 152–167. Springer-Verlag, 1992.

[BIG94] Jose L. Blanco, Arantza Illarramendi, and Alfredo Goñi. Building a federated relationale database system: An approach using a knowledge-based system. *Journal of Intelligent and Cooperative Information Systems*, 3(4):415–455, 1994.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Sham B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[BN97] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. In *Proc. of the 1997 Description Logic Workshop (DL-97)*, pages 34–38, 1997.

[Bor95] Alexander Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[CDGL95] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD-95)*, number 1013 in Lecture Notes in Computer Science, pages 229–246. Springer-Verlag, 1995.

[CDGL97a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive query containment in Description Logics with n-ary relations. In *Proc. of the 1997 Description Logic Workshop (DL-97)*, pages 5–9, 1997.

[CDGL97b] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Representing and reasoning on SGML documents. In *Proc. of the 10th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-97)*, 1997.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of IPSI Conference*, Tokyo (Japan), 1994.

[CHS91] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, 1991.

[CK86] S. S. Cosmadakis and P. C. Kanellakis. Functional and inclusion dependencies - A graph theoretical approach. In P. C. Kanellakis and F. P. Preparata, editors, *Advances in Computing Research, Vol. 3*, pages 163–184. JAI Press, 1986.

[CL93] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[CL94a] Diego Calvanese and Maurizio Lenzerini. Making object-oriented schemas more expressive. In *Proc. of the 13th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-94)*, pages 243–254, Minneapolis, 1994. ACM Press and Addison Wesley.

[CL94b] Diego Calvanese and Maurizio Lenzerini. On the interaction between ISA and cardinality constraints. In *Proc. of the 10th IEEE Int. Conf. on Data Engineering (ICDE-94)*, pages 204–213, Houston (Texas), 1994. IEEE Computer Society Press.

[CLN94] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109–120, Bonn, 1994. Morgan Kaufmann, Los Altos.

[CLN97] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Logics for data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*. Kluwer Academic Publisher, 1997. To appear.

[CPL97] Marco Cadoli, Luigi Palopoli, and Maurizio Lenzerini. Datalog and description logics: Expressive power. In *Proc. of the 6th Int. Workshop on Database Programming Languages (DBPL-97)*, 1997.

[DGL94a] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[DGL94b] Giuseppe De Giacomo and Maurizio Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proc. of the 4th European Workshop on Logics in Artificial Intelligence (JELIA-94)*, volume 838 of *Lecture Notes in Artificial Intelligence*, pages 332–346. Springer-Verlag, 1994.

[DGL95] Giuseppe De Giacomo and Maurizio Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, pages 801–807, 1995.

[DGL96] Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox reasoning in expressive description logics. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, editors, *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann, Los Altos, 1996.

[DGL97] Giuseppe De Giacomo and Maurizio Lenzerini. A uniform framework for concept definitions in description logics. *Journal of Artificial Intelligence Research*, 6:87–110, 1997.

[DGM96] Giuseppe De Giacomo and Fabio Massacci. Tableaux and algorithms for propositional dynamic logic with converse. In Michael A. McRobbie and John K. Slaney, editors, *Proc. of the 13th Int. Conf. on Automated Deduction (CADE-96)*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 613–628. Springer-Verlag, 1996.

[DLNS91] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. A hybrid system integrating Datalog and concept languages. In *Proc. of the 2nd Conf. of the Italian Association for Artificial Intelligence (AI*IA-91)*, number 549 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1991. An extended version appeared also in the Working Notes of the AAAI Fall Symposium "Principles of Hybrid Reasoning".

[DLNS96] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[DW91] David M. Dilts and Wenhua Wu. Using knowledge-based technology to integrate CIM databases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):237–245, 1991.

[Fer91] S. Ferg. Cardinality concepts in entity-relationship modeling. In *Proc. of the 10th Int. Conf. on the Entity-Relationship Approach (ER-91)*, pages 1–30, 1991.

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[GJM96] A. Gupta, H. V. Jagadish, and I. S. Mumick. Data integration using self-maintainable views. In *Proc. of the 5th Int. Conf. on Extending Database Technology (EDBT-96)*, number 1057 in Lecture Notes in Computer Science, pages 140–144. Springer-Verlag, 1996.

[GLN92] Willi Gotthard, Peter C. Lockemann, and Andrea Neufeld. System-guided view integration for object-oriented databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(1):1–22, 1992.

[GM84] John Grant and Jack Minker. Numerical dependencies. In H. Gallaire, J. Minker, and J.-M. Nicolas, editors, *Advances in Database Theory II*. Plenum Publ. Co., New York, 1984.

[GPC+92] J. Geller, Y. Perl, P. Cannata, A.P. Sheth, and E. Neuhold. A case study of structural integration. In *Proc. of the Int. Conf. on Information and Knowledge Management (CIKM-92)*, pages 102–111, 1992.

[GPNS92] J. Geller, Y. Perl, E. Neuhold, and A.P. Sheth. Structural schema integration with full and partial correspondence using the dual model. *Information Systems*, 17(6):443–464, 1992.

[GSSC95a] Manuel García-Solaco, Felix Saltor, and Malú Castellanos. A semantic-discriminated approach to integration in federated databases. In *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS-95)*, pages 19–31, 1995.

[GSSC95b] Manuel García-Solaco, Felix Saltor, and Malú Castellanos. A structure based schema integration methodology. In *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE-95)*, pages 505–512, 1995.

[HGMW+95] Joachim Hammer, Hector Garcia-Molina, Jennifer Widom, Wilburt Labio, and Yue Zhuge. The Stanford data warehousing project. *IEEE Bulletin of the Technical Committee on Data Engineering*, 18(2):41–48, 1995.

[HJK+93] Michael N. Huhns, Nigel Jacobs, Tomasz Ksiezyk, Wei-Min Shen an Munindar P. Singh, and Philip E. Cannata. Integrating enterprise information models in Carnot. In *Proc. of the Int. Conf. on Cooperative Information Systems (CoopIS-93)*, pages 32–42, 1993.

[Hor97] Ian Horrocks. Optimisation techniques for expressive description logics. Technical Report UMCS-97-2-1, University of Manchester, Department of Computer Science, 1997.

[Hul86] Richard Hull. Relative information capacity of simple relational database schemas. *SIAM Journal on Computing*, 15(3):856–886, 1986.

[Hul97] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-97)*, 1997.

[HZ96] Richard Hull and Gang Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 481–492, 1996.

[Inm96] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, second edition, 1996.

[Joh94] Paul Johanneson. Schema standardization as an aid in view integration. *Information Systems*, 19(3):275–290, 1994.

[KLK91] R. Krishnamurthy, W. Litwin, and W. Kent. Language features for interoperability of databases with schematic discrepancies. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 40–49, 1991.

[KLSS95] Thomas Kirk, Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

[KT90] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science – Formal Models and Semantics*, pages 789–840. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[LG90] Doug Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison Wesley Publ. Co., Reading, Massachussetts, 1990.

[LN90] Maurizio Lenzerini and Paolo Nobili. On the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems*, 15(4):453–461, 1990.

[LR96] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pages 323–327, 1996.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI-96)*, pages 40–47, 1996.

[LSK95] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5:121–143, 1995.

[MYR94] R. J. Miller, Y. E. Yoannidis, and R. Ramakrishnan. Schema equivalence in heterogeneous systems: Bridging theory and practice. *Information Systems*, 19(1):3–31, 1994.

[ON94] A. M. Ouksel and C. F. Naiman. Coordinating context building in heterogeneous information systems. *Journal of Intelligent Information Systems*, 3:151–183, 1994.

[Pel91] C. Peltason. The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119, 1991.

[PMB+91] Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representatin system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.

[Pra79] V. R. Pratt. Models of program logic. In *Proc. of the 20nd Annual Sym. on the Foundations of Computer Science (FOCS-79)*, pages 115–122, San Juan, 1979.

[Pra80] V. R. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20:231–255, 1980.

[Qia96] Xiaolei Qian. Correct schema transformations. In P. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. of the 5th Int. Conf. on Extending Database Technology (EDBT-96)*, number 1057 in Lecture Notes in Computer Science, pages 114–128, Avignon (France), 1996. Springer-Verlag.

[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(6):920–933, 1994.

[Sch91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, Australia, 1991.

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

[SPD92] Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model independent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1:81–126, 1992.

[Sti92] C. Stirling. Modal and temporal logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 477–563. Clarendon Press, Oxford, 1992.

[Tha92] Bernhard Thalheim. Fundamentals of cardinality constraints. In G. Pernoul and A. M. Tjoa, editors, *Proc. of the 11th Int. Conf. on the Entity-Relationship Approach (ER-92)*, pages 7–23. Springer-Verlag, 1992.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT-97)*, Lecture Notes in Computer Science, pages 19–40. Springer-Verlag, 1997.

[VW86] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC-84)*.

[VW94] V. Vidal and M. Winslett. Preserving update semantics in schema integration. In *Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM-94)*, pages 263–271, 1994.

[WGL+96] Janet L. Wiener, Himanshu Gupta, Wilburt J. Labio, Yue Zhuge, Hector Garcia-Molina, and Jennifer Widom. A system prototype for warehouse view maintenance. Technical report, Stanford University, 1996. Available at http://www-db-stanford.edu/warehousing/warehouse.html.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, March 1992.

[YPS94] Xian Ye, Christine Parent, and Stefano Spaccapietra. Cardianlity consistency of derived objects in DOOD systems. In P. Loucopoulos, editor, *Proc. of the 13th Int. Conf. on the Entity-Relationship Approach (ER-94)*, number 881 in Lecture Notes in Computer Science, pages 278–295, Manchester (UK), 1994. Springer-Verlag.

[ZGMHW95] Yue Zhuge, Hector Garcia-Molina, Joachim Hammer, and Jennifer Widom. View maintenance in a warehousing environment. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 316–327, 1995.

[ZHK96] Gang Zhou, Richard Hull, and Roger King. Generating data integration mediators that use materializations. *Journal of Intelligent Information Systems*, 6:199–221, 1996.

[ZHKF95a] Gang Zhou, Richard Hull, Roger King, and Jean-Claude Franchitti. Data integration and warehousing using H20. *IEEE Bulletin of the Technical Committee on Data Engineering*, 18(2):29–40, 1995.

[ZHKF95b] Gang Zhou, Richard Hull, Roger King, and Jean-Claude Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS-95)*, pages 4–18, 1995.

Appendices

We include as appendices the following publications: [CDGL97a, CLN97, CDGL97b, DGL97, BN97].