Extracting Ontologies from Relational Databases

Lina Lubyte and Sergio Tessaris

Faculty of Computer Science – Free University of Bozen-Bolzano

1 Introduction

The use of a conceptual model or an ontology over data sources has been shown to be necessary to overcome many important database problems (for a survey see [1]). Since ontologies provide a conceptual view of the application domain, the recent trend to employ such ontologies for navigational (and reasoning) purposes when accessing the data gives additional motivation for the problem of extracting the ontology from database schema [2]. When such an ontology exists, modelling the relation between the data sources and an ontology is a crucial aspect in order to capture the semantics of the data.

In this paper we define the framework for extracting from a relational database an ontology that is to be used as a conceptual view over the data, where the semantic mapping between the database schema and the ontology is captured by associating a view over the source data to each element of the ontology. Thus, the vocabulary over the ontology can be seen as a set of (materialised) views over the vocabulary of the data source; i.e., a technique known as GAV approach in the information integration literature [3]. To describe the extracted conceptual model, we provide an expressive ontology language which can capture features from Entity-Relationship and UML class diagrams, as well as variants of Description Logics. The heuristics underlying the ontology extraction process are based on ideas of standard relational schema design from ER diagrams in order to uncover the connections between relational constructs and those of ontologies. Besides the latter assumption the procedure presented in this paper takes into consideration relations being in third normal form (3NF). Under this assumption we can formally prove that the conversion preserves the semantics of the constraints in the relational database. Therefore, there is no data loss, and the extracted model constitutes a faithful wrapper of the relational database.

2 Preliminaries

We assume that the reader is familiar with standard relational database notions as presented, for example, in [4]. We assume that the database domain is a fixed denumerable set of elements Δ and that every such element is denoted uniquely by a constant symbol, called its *standard name* [5]. We make use of the standard notion of relational model by using named attributes, each with an associated datatype, instead of tuples.

A *relational schema* $\mathcal{R}$ is a set of relationships, each one with a fixed set of attributes (assumed to be pairwise distinct) with associated datatypes. We

use $[s_1 : D_1, \ldots, s_n : D_n]$ to denote that a relationship has attributes $s_1, \ldots, s_n$ with associated data types $D_1, \ldots, D_n$. We interpret relationships over a fixed countable *domain* Δ of datatype elements, which we consider partitioned into the datatypes D_i. A *database instance* (or simply *database*) $\mathcal{D}$ over a relational schema $\mathcal{R}$ is an (interpretation) function that maps each relationship R in $\mathcal{R}$ into a set $R^{\mathcal{D}}$ of total functions from the set of attributes of R to Δ. Let $A = [s_1, \ldots, s_m]$ be a sequence of m attribute names of a relationship R of a schema $\mathcal{R}$. The *projection* of $R^{\mathcal{D}}$ over A is the relation $\pi_A R^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^m$, satisfying the condition that $\phi \in R^{\mathcal{D}}$ iff $(\phi(s_1), \ldots, \phi(s_m)) \in \pi_A R^{\mathcal{D}}$.

The ontology extraction task takes as input a relational source; e.g. a DBMS. We abstract from any specific database implementation by considering an abstract *relational source* $\mathcal{DB}$, which is a pair $(\mathcal{R}, \Sigma)$, where $\mathcal{R}$ is a relational schema and Σ is a set of *integrity constraints*. The semantics of relational schemata is provided in the usual way by means of the relational model. Below we briefly list the kind of database integrity constraints we consider in our framework (for more details the reader is referred to [6]). *Nulls-not-allowed constraints*: satisfied in a database when null are not contained in any indicated attribute. *Unique constraints*: satisfied when the sequence of attributes are unique in a relation. Together with nulls-not-allowed constraints they correspond to *key constraints*. *Inclusion dependencies*: satisfied when the projection of two relations are included one in the other. When the attributes of the target relation are a candidate key as well, we call them *foreign key* constraints. *Exclusion dependencies*: satisfied when the intersection of the projection of two relations is the empty set. *Covering constraints*:[1] between a relation and a set of relations, satisfied when the projection of the relation over the specified attributes is included in the union of the projections of the relations in the set.

We call a *DLR-DB* system $\mathcal{S}$ a triple $\langle \mathcal{R}, \mathcal{P}, \mathcal{K} \rangle$, where $\mathcal{R}$ is a *relational schema*, $\mathcal{P}$ is a *component structure* over $\mathcal{R}$, and $\mathcal{K}$ is a set of assertions involving names in $\mathcal{R}$. The intuition behind a named component is the role name of a relationship in an ER schema (or UML class-diagram). The *component structure* $\mathcal{P}$ associates to each relationship a mapping from *named components* to sequences of attributes. Let R be a relationship in $\mathcal{R}$, to ease the notation we write $\mathcal{P}_R$ instead of $\mathcal{P}(R)$.

Let R be a relationship in $\mathcal{R}$, with attributes $[s_1 : D_1, \ldots, s_n : D_n]$. $\mathcal{P}_R$ is a non-empty (partial) function from a set of named components to the set of nonempty sequences of attributes of R. The domain of $\mathcal{P}_R$, denoted $\mathcal{C}_R$, is called the set of *components of* R. For a named component $c \in \mathcal{C}_R$, the sequence $\mathcal{P}_R(c) = [s_{i_1}, \ldots, s_{i_m}]$, where each $i_j \in \{1, \ldots, n\}$, is called the *c-component* of R. We require that the sequences of attributes for two different named components are not overlapping, and that each attribute appears at most once in each sequence. The *signature* of a component $\mathcal{P}_R(c)$, denoted $\tau(\mathcal{P}_R(c))$, is the sequence of types of the attributes of the component. Two components $\mathcal{P}_R(c_1)$ and $\mathcal{P}_R(c_2)$ are *compatible* if the two signatures $\tau(\mathcal{P}_R(c_1))$ and $\tau(\mathcal{P}_R(c_2))$ are equal.

[1] In ER terminology, this may also be indicated as *mandatory* for an IS-A relationship.

The *DLR-DB* ontology language, used to express the assertions in $\mathcal{K}$, is based on the idea of modelling the domain by means of *axioms* involving the projection of the relationship over the named components. An *atomic formula* is a projection of a relationship R over one of its components. The projection of R over the c-component is denoted by $R[c]$. When the relationship has a single component, then this can be omitted and the atomic formula R corresponds to its projection over the single component. Given the atomic formulae $R[c], R'[c'], R_i[c_i]$, an *axiom* is an assertion of the form specified below; where all the atomic formulae involved in the same axiom must be compatible. The semantics is provided in terms of relational models for $\mathcal{R}$, where $\mathcal{K}$ plays the role of constraining the set of "admissible" models.

$$R[c] \sqsubseteq R'[c'] \;\; \pi_c R^{\mathcal{D}} \subseteq \pi_{c'} R'^{\mathcal{D}} \qquad \text{Subclass}$$

$$R[c] \text{ disj } R'[c'] \;\; \pi_c R^{\mathcal{D}} \cap \pi_{c'} R'^{\mathcal{D}} = \emptyset \qquad \text{Disjointness}$$

$$\text{funct}(R[c]) \text{ for all } \phi_1, \phi_2 \in R^{\mathcal{D}} \text{ with } \phi_1 \neq \phi_2, \text{ we have } \phi_1(s) \neq \phi_2(s) \text{ for some } s \text{ in } c \qquad \text{Functionality}$$

$$R_1[c_1], \ldots, R_k[c_k] \text{ cover } R[c] \;\; \pi_c R^{\mathcal{D}} \subseteq \bigcup_{i=1\ldots k} \pi_{c_i} R_i'^{\mathcal{D}} \qquad \text{Covering}$$

A database $\mathcal{D}$ is said to be a *model* for $\mathcal{K}$ if it satisfies all its axioms, and for each relationship R in $\mathcal{R}$ with components $c_1, \ldots, c_k$, for any $\phi_1, \phi_2 \in R^{\mathcal{D}}$ with $\phi_1 \neq \phi_2$, there is some s in c_i s.t. $\phi_1(s) \neq \phi_2(s)$. The above conditions are well defined because we assumed the compatibility of the atomic formulae involved in the constraints.

The *DLR-DB* ontology language enables the use of the most commonly used constructs in conceptual modelling (see [7]). Note that by taking away the covering axioms and considering only components containing single attributes this ontology language corresponds exactly to *DLR-Lite* (see [8]). By virtue of the assumption that components do not share attributes, it is not difficult to show that the same reasoning mechanism of *DLR-Lite* can be used in our case. The discussion on the actual reasoning tasks which can be employed in the context of *DLR-DB* systems is out of the scope of this paper. Herewith we are mainly interested of the use of the language to express data models extracted from the relational data sources.

3 Ontology Extraction

Our proposed ontology extraction algorithm works in two phases. Firstly, a classification scheme for relations from the relational source is derived. Secondly, based on this classification, the ontology describing the data source is extracted. Moreover, the process generates a set of view definitions, expressing the mapping between the database schema and the ontology. In this section we briefly sketch the procedure, more details and the algorithms can be found in [6].

The principles upon our technique are based on best practices on relational schema design from ER diagrams – a standard database modelling technique [9]. One benefit of this approach is that it can be shown that our algorithm, though heuristic in general, is able to reconstruct the original ER diagram under some

assumptions on the latter. Specifically, we consider ER models that support entities with attributes,[2] n-ary relationships which are subject to cardinality constraints, and inheritance hierarchies (IS-A) between entities (including multiple inheritance) which may be constrained to be disjoint or covering. Roughly speaking, we reverse the process of translating ER model to relational model. As a result, we identify that relations representing entities have keys which are not part of their foreign keys, and every such foreign key represents functional binary relationship (i.e., one-to-one or one-to-many) with another entity. On the other hand, relations that correspond to n-ary relationships with cardinalities "many" for all participating entities have keys composed of their foreign keys. Since we assume there is no IS-A between relationships, every such foreign key references key of a relation resulting from (sub-)entity. When a relation has key that is also its foreign key, and no other non-key foreign keys appear in that relation, then, clearly, an inheritance relationship exists. If instead non-key foreign keys are present but the relation is the target of some foreign key, we are sure that this relation corresponds to sub-entity. Otherwise, such relation might also "look like" functional relationship (binary or n-ary), mapped directly to a relation, and therefore relations of this type are classified as ambiguous relations (see below).

The classification of the relations, based on their keys and foreign keys, can be summarised as: *base relation* when the primary key is disjoint with every foreign key; *specific relation* when the primary key is also a foreign key and it has a single foreign key, or it is referred to by some relation;[3] *relationship relation* when the primary key is composed by all the foreign keys, which are more than 1. Any other relation is classified as *ambiguous*.

Once the relations in $\mathcal{DB}$ are classified according to the conditions defined above, then the actual ontology extraction process returns a *DLR-DB* system as output. For every base and specific relation r_i the algorithm generates a relationship R_i with the attributes in a one-to-one correspondence with non-foreign key attributes of r_i, and a single c-component, where $\mathcal{P}_{R_i}(c)$ corresponds to the key attributes of r_i and thus functionality axiom $\mathrm{funct}(R_i[c])$ is added to $\mathcal{K}$;a view is defined by projecting on all non-key foreign key attributes of r_i.

Once relationships for base and specific relations are defined, associations between those relationships must be identified. Specifically, a non-key foreign key in a relation r_i referencing relation, r_j, determines the association between relationships R_i and R_j. Thus, for each such foreign key, a relationship R_k is generated, having two components, c_i-component and c_j-component, where $\mathcal{P}_{R_k}(c_i)$ and $\mathcal{P}_{R_k}(c_j)$ correspond to key attributes of r_i and r_j, respectively, where c_i-component is functional, i.e., we have $\mathrm{funct}(R_k[c_i])$ in $\mathcal{K}$; a corresponding view is defined by joining r_j with r_i and projecting on their keys. For expressing an association, determined by R_k, between R_i and R_j the axioms of the form $R_k[c_i] \sqsubseteq R_i$ and $R_k[c_j] \sqsubseteq R_j$ are added to $\mathcal{K}$. Furthermore, whenever the latter foreign key of r_i participates in a nulls-not-allowed constraint, the axiom $R_i \sqsubseteq R_k[c_i]$ is generated stating mandatory participation for instances of R_i to

[2] We do not deal with multi-valued attributes in this paper.

[3] I.e., the relation appears on the right-hand side of some foreign key constraint.

R_k as values for the c_i-component; its participation to a unique constraint determines instead the functionality axiom funct($R_k[c_j]$) meaning that every value of the c_j-component appears in it at most once; finally, appearance of the foreign key of r_i in the right-hand side of an inclusion dependency determines mandatory participation for values of the only component of R_j to the relationship R_k as values for the c_j-component, and thus the axiom $R_j \sqsubseteq R_k[c_j]$ is added to $\mathcal{K}$. For expressing an ISA between classes, for every specific relation r_i the subclass axiom $R_i \sqsubseteq R_j$ is added to $\mathcal{K}$, where R_j is the relationship corresponding to (base or specific) relation, r_j, that the key foreign key of r_i references. Additionally, each exclusion dependency on the set of specific relations induces the disjointness axioms R_i disj R_k, for every pair of relations r_i, r_k appearing in the exclusion dependency. Similarly, every covering constraint on the set of specific relations induces the corresponding covering axiom in $\mathcal{K}$.

Each relationship relation r_i is accounted for by generating a relationship R_i, with attributes in a one-to-one correspondence with those of r_i, and n components, where n is the number of foreign keys of r_i. Each $\mathcal{P}_{R_i}(c_{i_l})$ ($l \in \{1, \ldots, n\}$) has sequence of attributes corresponding to the l-th foreign key attributes of r_i; the corresponding view is defined by projecting on all attributes of r_i. Then, for each foreign key of r_i referencing relation r_j (that is already represented with a relationship R_j having a single component), the algorithm generates an axiom $R_i[c_{i_l}] \sqsubseteq R_j$ stating that the role corresponding to the c_{i_l}-component of R_i is of type R_j. Furthermore, if this foreign key appears on the right-hand side of an inclusion dependency, the axiom $R_j \sqsubseteq R_i[c_{i_l}]$ is added to $\mathcal{K}$ that states mandatory participation for instances of R_j to the relationship R_i as values for the c_{i_l} component.

Finally, the appropriate structures for ambiguous relations must be identified. As already discussed before, an ambiguous relation may correspond in ER schema to either sub-entity, which also participates with cardinality "one" in a binary relationship, or a functional relationship that was directly mapped to a relation. Following the idea that all functional binary relationships should be represented in a relational model with an embedded foreign key, e.g., in order to obtain the relational schema with a minimum number of relations, and that n-ary relationships ($n \geq 3$) are relatively unusual, our heuristics "prefers" to recover an inheritance relationship, and thus the algorithm generates the structures corresponding to those defined for specific relations. On the other hand, a user could decide which is the "best" structure for ambiguous relations. In this way, the ontology extraction task may be a completely automated procedure, or semi-automated process with a user intervention.

As an example of the ontology extraction process, consider the relational schema (primary keys are underlined) with constraints of Figure 1. At the initial step of extraction process, relations Scholar, Publication and Department are classified as base relations, i.e. their keys and foreign keys do not share any attributes; IsAuthorOf relation is classified as relationship relation – its key is entirely composed from foreign keys; while relations PostDoc and Professor satisfy the conditions required for specific relations, i.e. the key ssn is their single foreign key.

Without going into details of the algorithm, in Table 1 we list the extracted relationships of *DLR-DB* $\mathcal{R}$ together with the devised component structure $\mathcal{P}$, by considering the relation names and their corresponding attributes in the input relational source. Starting with base and specific relations, we have the corresponding relationships with single components. Since the component names for the latter relationships are not relevant (they can be omitted), we choose a common name id for all the five of them.[4] Figure 2 shows the extracted ontology together with the corresponding ER diagram.

Relationship	**Component** c	$\mathcal{P}_R(c)$	**Additional attr.**	**View definition**
Scholar	id	ssn	name	$\pi_{\text{ssn,name}}(\text{Scholar}_r)$
Publication	id	id	title, year	$\pi_{\text{id,title,year}}(\text{Publication}_r)$
Department	id	no	name	$\pi_{\text{no,name}}(\text{Department}_r)$
PostDoc	id	ssn	scholarship	$\pi_{\text{ssn,scholarship}}(\text{PostDoc}_r)$
Professor	id	ssn	salary	$\pi_{\text{ssn,salary}}(\text{Professor}_r)$
WorksFor	employee	ssn		$\pi_{\text{ssn,no}}(\text{Department}_r \bowtie \text{Scholar}_r)$
	dept	no		
IsAuthorOf	author	schSsn		$\pi_{\text{schSsn,publId}}(\text{IsAuthorOf}_r)$
	publication	publId		

Table 1. Extracted Schema.

4 Discussion and Related Work

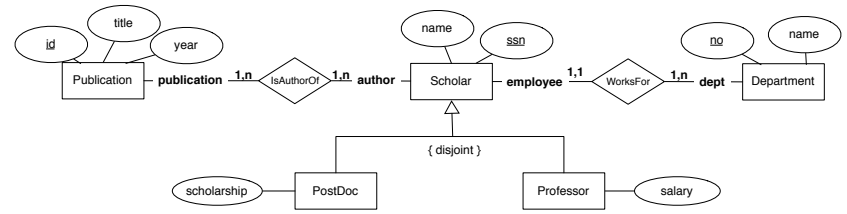
Much work has been addressed on the issue of explicitly defining semantics in database schemas [10, 7] and extracting semantics out of database schemas [11, 12]. The work described in [10] provides algorithms that investigate data instances of an existing legacy database in order to identify candidate keys of

[4] For the sake of clarity, the naming of the components for relationships WorksFor and IsAuthorOf, as well as the name of the WorksFor relationship itself, are determined by domain knowledge.

$\text{Scholar}_r(\underline{\text{ssn}}, \text{name}, \text{deptNo})$
$\text{IsAuthorOf}_r(\underline{\text{schSsn}, \text{publId}})$
$\text{PostDoc}_r(\underline{\text{ssn}}, \text{scholarship})$
$\text{Publication}_r(\underline{\text{id}}, \text{title}, \text{year})$
$\text{Department}_r(\underline{\text{no}}, \text{name})$
$\text{Professor}_r(\underline{\text{ssn}}, \text{salary})$

$\text{Scholar}_r\,[\text{deptNo}] \subseteq \text{Department}_r\,[\text{no}]$
$\text{IsAuthorOf}_r\,[\text{schSsn}] \subseteq \text{Scholar}_r\,[\text{ssn}]$
$\text{IsAuthorOf}_r\,[\text{publId}] \subseteq \text{Publication}_r\,[\text{id}]$
$\text{PostDoc}_r\,[\text{ssn}] \subseteq \text{Scholar}_r\,[\text{ssn}]$
$\text{Professor}_r\,[\text{ssn}] \subseteq \text{Scholar}_r\,[\text{ssn}]$
$\text{Scholar}_r\,[\text{ssn}] \subseteq \text{IsAuthorOf}_r\,[\text{schSsn}]$
$\text{Publication}_r\,[\text{id}] \subseteq \text{IsAuthorOf}_r\,[\text{publId}]$
$\text{Department}_r\,[\text{no}] \subseteq \text{Scholar}_r\,[\text{deptNo}]$
$\mathit{unique}(\text{Scholar}_r, \text{deptNo})$
$\mathit{nonnull}(\text{Scholar}_r, \text{deptNo})$
$\text{PostDoc}_r\,[\text{ssn}] \cap \text{Professor}_r\,[\text{ssn}] = \emptyset$

Fig. 1. Relational schema with constraints.



funct(Scholar[id])
funct(Publication[id])
funct(Department[id])
funct(PostDoc[id])
funct(Professor[id])
IsAuthorOf[author] $\sqsubseteq$ Scholar
IsAuthorOf[publication] $\sqsubseteq$ Publication
WorksFor[employee] $\sqsubseteq$ Scholar
WorksFor[dept] $\sqsubseteq$ Department

Scholar $\sqsubseteq$ IsAuthorOf[author]
Publication $\sqsubseteq$ IsAuthorOf[publication]
Scholar $\sqsubseteq$ WorksFor[employee]
Department $\sqsubseteq$ WorksFor[dept]
funct(WorksFor[employee])
PostDoc $\sqsubseteq$ Scholar
Professor $\sqsubseteq$ Scholar
PostDoc disj Professor

Fig. 2. Extracted ontology and corresponding ER diagram.

relations, to locate foreign keys, and to decide on the appropriate links between the given relations. As a result, user involvement is always required. In our work we instead assume the knowledge on key and foreign key constraints, as well as non null and unique values on attributes, inclusion and disjointness between relations, etc. exist in the schema.

The work in [12] propose transformations that are applied to produce the re-engineered schema and handles the establishment of inheritance hierarchies. However, it considers relations in BCNF and thus every relation is in a one-to-one correspondence with an object in the extracted schema. The main idea of the methodology described in [11] comes close to ours in the sense that it derives classification for relations and attributes based on heuristics of what kind of ER components would give rise to particular relations. Unlike the latter, our proposed technique can be seen as a *schema transformation* as defined in [15]. Because of lack of space, we omit the proof that this transformation is indeed *equivalence preserving* (for the actual proof and details see [6]).

The recent call for a Semantic Web arose several approaches in bringing together relational databases and ontologies. Among them we mention [13], where the authors describe an automatic mapping between relations and ontologies, when given as input simple correspondences from attributes of relations to datatype properties of classes in an ontology. Unlike our approach, it requires a target ontology onto which the relations are mapped to. On the other hand, the approach of [14] extracts the schema information of the data source and converts it into an ontology. However, the latter technique extracts only the structural information about the ontology, so the constraints are not taken into account.

This paper describes an heuristic procedure for extracting from relational database its conceptual view, where the wrapping of relational data sources by means of an extracted ontology is done by associating view over the original data to each element of the ontology. To represent the extracted ontology, instead of

a graphical notation, we employ an ontology language thus providing a precise semantics to extracted schema. Our extraction procedure relies on information from the database schema and automatically extracts all the relevant semantics if an input relational schema was designed using a standard methodology.

We are currently following several directions to continue the work reported in this paper. First, conceptual modelling constructs as multi-valued attributes and weak entities, alternative techniques for inheritance representation in relational tables. To this purpose we are starting to experiment with real database schemas to evaluate the quality of the extracted ontologies.

References

1. Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hubner, S.: Ontology-based integration of information - a survey of existing approaches. In: Proc. of IJCAI-01 Workshop: Ontologies and Information Sharing. (2001) 108–117
2. Lembo, D., Lutz, C., Suntisrivaraporn, B.: Tasks for ontology design and maintenance. Deliverable D05, TONES EU-IST STREP FP6-7603 (2006)
3. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of PODS02. (2002) 233–346
4. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. (1995)
5. Levesque, H.J., Lakemeyer, G.: The Logic of Knowledge Bases. The MIT Press (2001)
6. Lubyte, L., Tessaris, S.: Extracting ontologies from relational databases. Technical report, Free University of Bozen-Bolzano (2007) Available at http://www.inf.unibz.it/krdb/pub/index.php.
7. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on uml class diagrams. Artificial Intelligence **168**(1) (2005) 70–118
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR 2006. (2006) 260–270
9. Batini, C., Ceri, S., Navathe, S.B.: Conceptual Database Design. An Entity-Relationship Approach. Benjamin/Cummings Publishing Company, Inc. (1992)
10. Alhajj, R.: Extracting an extended entity-relationship model from a legacy relational database. Information Systems **26**(6) (2003) 597–618
11. Chiang, R.H.L., Barron, T.M., Storey, V.C.: Reverse engineering of relational databases: extraction of an eer model from a relational database. Data and Knowledge Engineering **12**(2) (1994) 107–142
12. Markowitz, V.M., Makowsky, J.A.: Identifying extended entity-relationship object structures in relational schemas. IEEE Transactions on Software Engineering **16**(8) (1990) 777–790
13. An, Y., Borgida, A., Mylopoulos, J.: Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In: ODBASE'05. (2005) 1152–1169
14. Laborda, C.P., Conrad, S.: Bringing relational data into the semantic web using sparql and relational.owl. In: Proc. of the 22nd Int. Conf. on Data Engineering Workshops (ICDEW'06). (2006) 55–62
15. Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In: Proc. of VLDB'93, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1993) 120–133