Query Processing Using Ontologies

Chokri Ben Necib and Johann-Christoph Freytag

Humboldt-Universität zu Berlin, Germany
{necib, freytag}@dbis.informatik.hu-berlin.de

Abstract. Recently, the database and AI research communities have paid increased attention to *ontologies*. The main motivating reason is that ontologies promise solutions for complex problems caused by the lack of a good understanding of the semantics of data in many cases. In particular, ontologies have extensively been used to overcome the interoperability problem during the integration of heterogeneous information sources. Moreover, many efforts have been put into developing ontology based techniques for improving the query answering process in database and information systems.

In this paper, we present a new approach for query processing within single (object) relational databases using ontology knowledge. Our goal is to process database queries in a semantically more meaningful way. In fact, our approach shows how an ontology can be effectively exploited to rewrite a user query into another one such that the new query provides more meaningful results satisfying the intention of the user. To this end, we develop a set of transformation rules which rely on semantic information extracted from the ontology associated with the database. In addition, we propose a semantic model and a set of criteria to prove the validity of the transformation results. We also address the necessary mappings between an ontology and its underlying database w.r.t. our framework.

1 Introduction

With the rapid growth of data in databases and information sources and the increasing demands for exchanging information through the internet, the challenges in accessing data become more complex than in past few decades. The major problems are: (i) hiding the heterogeneity in format and structure of data from the users, (ii) overcoming the confusion in terminologies caused by employing synonyms and homonyms, and (iii) providing users with the most relevant answers to his requests in less time and/or resources. Therefore, the need to "understand" data of the information sources is increasing. Web search engines, for example, try to replace their syntactic based retrieval of information by a semantic based one [5]. In this context, researchers become aware of the usefulness of semantic knowledge to deal with the problems above. Indeed, semantic knowledge about a specific source can be considered as a meta-data layer over the instances of the underlying source.

Recently, *ontologies* have become popular candidates to capture such semantics. The reason is that an ontology can provide a shared common understanding of the application domain in concise and consensual manners. In fact, ontologies provide the

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 167–186, 2005.

meaning of terms and their relationships by which the domain is modelled [20]. They have been proven to be an important support for managing data in database and information systems for overcoming the interoperability problem of heterogeneous information sources. Thus, users should not care about where and how the data are organized in the sources. For this reason, in systems like OBSERVER [12] and TAMBIS [16] users formulate their queries over a given ontology without directly accessing the data sources themselves. In the meantime, ontologies are also used to enhance the functionality of Web search engines by associating meaning with the content of Web pages. Several approaches propose to annotate Web resources with ontology knowledge and inference mechanisms to improve the search [19, 9]. These efforts, among others, converge to build the so called *Semantic Web*.

In this paper, we present a new approach on how to improve the answers of database queries based on semantic knowledge expressed in ontologies. Given a database, we assume the existence of an ontology which is associated with the database and which provides the context of its objects. We show how an ontology can be exploited effectively to reformulate a user query such that the new query can provide more "meaningful" answer meeting the intention of the user. A query can be defined by a set of selections and projections over database objects satisfying a set of conditions. These conditions are defined by a set of terms and determine the answer to the query. If a user wants to retrieve information from a database about certain objects, he might use terms, which do not exactly match the database values (due to the mismatch between the user's world view and the database designer's world view). However, there might be values in the database that are syntactically different from one another but semantically equivalent to the user terms and that express the same intention of the user. We address this issue as a semantic problem rather than as a pattern matching problem.

The remainder of this paper is organized as follows. First, we state the problem illustrating it by some examples. Then, the concepts of "ontologies" are described. In Section 4 we present our approach for query processing and propose necessary reformulation rules. In order to prove the soundness of the approach a semantic model and a set of criteria are proposed in Section 5. Mappings used to connect an ontology to its underlying database are discussed in Section 6. Finally, Section 7 concludes the paper.

2 Motivation and Problem Statement

Traditional techniques for query processing which rely on syntactic approaches become insufficient to cope with problems caused by the heterogeneity of data in its format and structure [22]. Current database and information systems require "more knowledge" about information sources in order to retrieve data in an efficient manner and satisfy the user expectations. For instance, semantic knowledge in the form of integrity constraints have been extensively used for developing query optimization techniques. There, the goal is to rewrite a user query into another query which can return the same result in less time and/or less resources [3]. This paper outlines a new approach for query processing which exploits data semantics in forms of ontologies to provide users with

Table 1. Item relation

A-ID	Name	Model	Price
123	computer	ibm	3000 $
124	intelPc	toshiba	5000 $
125	notebook	dell	4000 $
127	pc	compaq	2500 $
128	product	hp	3000 $
129	monitor	elsa	1000 $
135	keyboard	itt	80 $
136	desktop	ibm	1000 $
140	macPc	mac	2000 $
141	calculator	siemens	1500 $

Table 2. Component relation

S-ID	M-ID
123	129
123	135
123	136
124	129
124	135
124	136
125	135
127	129
127	135
127	136
128	129
128	135
128	136
140	129
140	135
140	136
141	135

meaningful answers to their queries. The basic idea is to allow the DBMS to deal with user queries both at the semantic as well as the syntactic level. There, users do not need to fully understand the database content to issue their queries and the resulting database answers could fulfil completely their expectations. In fact, if a user attempts to retrieve information about certain objects from a database, the answer to his query might not satisfy his needs. This can be justified by several facts. First, the information stored in databases are usually captured in natural languages. This leads to several variations in expression of the same concept (synonym problem). Moreover, languages introduce multiple meanings of the same expression (homonym problem). These problems might affect the query results when formulating queries using certain terms. Second, there might be different ways to formulate a query using semantically equivalent terms. We define two sets of terms to be semantically equivalent if they have the same meaning i.e. if their relevant concepts and relationships in the ontology identify the same concept. For example, if two terms are synonyms, they are semantically equivalent. There might be several such sets. Therefore, when a user formulates his query, he might use terms partially covering these semantics. Third, some results in the answer might not be related to the same context associated with the query. The context must be defined by the user. We consider the following example to illustrate these ideas throughout the paper.

Example 1: Assuming we have a relational database, denoted by DB_1. This database contains information about technical items of a store and includes two relations called 'Item' and 'Component': The relation Item contains a set of items described by the attributes 'name', 'model' and 'price'. The relation component contains the parts belonging to each item. The relational schema of DB_1 is described as follows:

ITEM(A-ID, Name, Model, Price)
A-ID: Item identifier
Name: Name of the Item
Model: Model of the Item
Price: Price of the Item
PrimaryKey(A-ID)

COMPONENT (S-ID, M-ID)
M-ID: Main part identifier
S-ID: Second part identifier
Foreign-Key(M-ID) TO ITEM
Foreign-Key(S-ID) TO ITEM
Primary-Key(S-ID,M-ID)

Suppose, at present, that DB_1 contains the instances as shown in the Tables 1 and 2. Querying the database DB_1 to retrieve information about the Item ”computer” also means information about the Items ”data processor” and ”calculator” because these terms are synonymous with the term ”computer”. Consequently, if a user formulates his query specifying only the term ”computer” he might miss other tuples concerning ”data processor” and ”calculator”. In addition ”computer” is implied by other terms which should be considered in the query. This example seems to be simple, but there could be more complicated ones depending on the nature of the query as we shall see later. In fact, the difference between the user's perception of real world objects and the database designer, who registers information about these objects, might cause semantic ambiguities including a ”vocabulary problem”. Therefore, it is hard for the DBMS to solve such semantic problems without additional semantic knowledge like ontologies.

In summary, we state our problem as follows:
Given a database DB, an ontology O and a user query Q, find a reformulated query Q' of Q by using O such that Q' returns more meaningful answer to the user than Q.

3 Ontology

3.1 Definition

In recent years, the term ”Ontology” has become a ”buzz word” for researches in the fields of databases and artificial intelligence. There are many definitions of what an ontology is [7, 8, 15, 4, 19]. An initial definition was given by Tom Gruber: ”*An ontology is an explicit specification of a conceptualization*” [7]. Ontologies have been increasingly emerging because of the crucial role that they play: Ontologies provide a concise and unambiguous description of concepts and their relationships for a domain of interest. This knowledge can be shared and reused by different participants.

Informally, we define an ontology as an intentional description of what is known about the essence of the entities in a particular domain of interest using abstractions, also called *concepts* and the *relationships* among them. Basically, the hierarchical organization of concepts through the inheritance (`"ISA"`) relationship constitutes the backbone of an ontology. Other kinds of relationship like part-whole (`"PartOf"`) or Synonym (`"SynOf"`) or application specific relationships might also exist. Furthermore, a set of logical axioms is often associated with the ontology to specify semantics of the relationships. To the best of our knowledge, there is no work until now addressing the issue of using ontology relationships at the database instance level.

For clarity, we have to distinguish between the meaning of the term ”concept” and that of the term ”concept instance”. A concept is a description of a group of real world objects in a certain domain whereas a concept instance is a set of values that represent these objects [2]. Note that many real-world ontologies already combine data instances and concepts [8]. In our definition we do not consider instances as part of an ontology.

For the remainder of the paper we refer to the set of the ontology concepts as $\zeta = \{c_1, \ldots, c_n\}$ and the set of ontological relationships as $\Re = \{\text{"}\texttt{ISA}\text{"}, \text{"}\texttt{SynOf}\text{"}, \text{"}\texttt{PartOf}\text{"}, \ldots\}$, where $c_i \in \zeta$ and $r_i \in \Re$ are non-null strings. We denote the set of axioms by $\Im$.

3.2 Graphical Representation

An ontology can be then represented as a directed labelled graph $G(V, E)$, where V is a finite set of vertices and E is a finite set of edges: Each vertex of V is labelled by a concept from ζ and each edge of E is labelled by an inter-concept relationship from $\Re$. Note that instances are not represented in G because they do not belong to an ontology. Further, we refer to a node by its label (a concept) and refer to an edge by its node concepts and its label (a relationship). For instance, the statement $e = c_1\, R_i\, c_2$ refers to the edge between the concept nodes c_1 and c_2 which is labelled by a relationship R_i. Formally, the graph G can be expressed as a relation $G \subseteq \zeta \times \Re \times \zeta$. Appendix A gives the most important graph operations that are used to extract concepts for query reformulations.

Figure 1 gives an example of a graph representation of a fragment of an ontology called "Product Ontolgy" (denoted by O_1). The ontology describes concepts and their

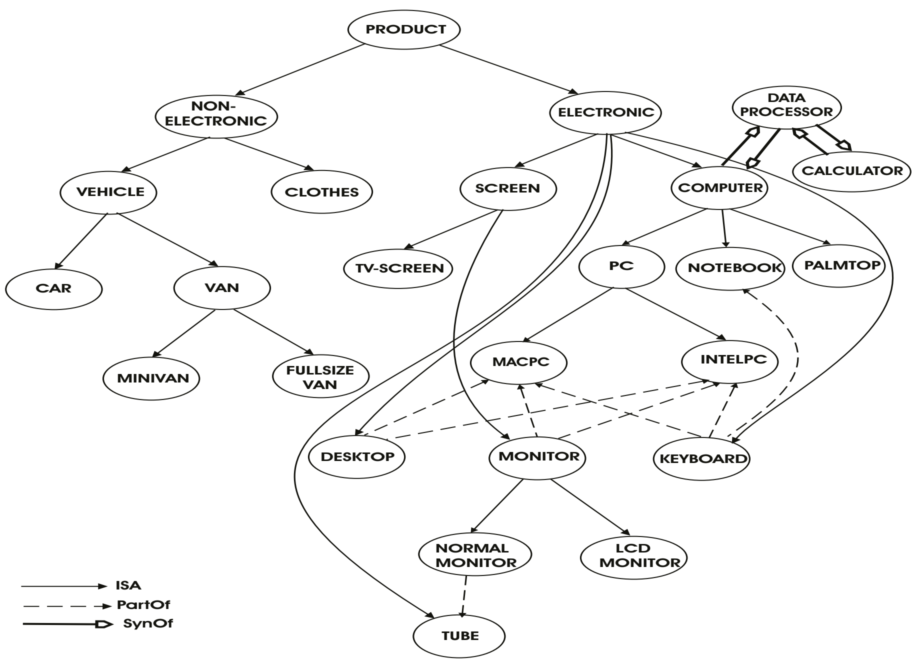

Fig. 1. Product Ontology O_1

relationships related to products. A part of this ontology is adopted from an ontology described in [11].

In summary, we define an ontology as the following set: $O = \{G, \zeta, \Re, \Im\}$.

4 Ontology Based Query Processing

The objective of our approach to query processing is to determine an alternative way to reformulate an input query into another "meaningful" query but not necessary equivalent one. The approach can be applied to the DBMS in a simple manner without any complex modifications of its core. Figure 2 shows an overview on the system's architecture. The system mainly consists of three components. The first component is the transformation engine which constitutes the core of the system. It performs a preprocessing of an input query, say Q, before submitting it to the database. This is done by reformulating Q into another query, say Q', in a semantic meaningful way using a set of semantic rules. These rules rely on additional semantics extracted from an ontology. Basically, they must contribute to:

- Expand user queries by changing their select conditions using synonyms for the terms in the condition and others specifying them.
- Substitute the query conditions with other conditions that are semantically equivalent.
- Reduce the scope of queries by restricting its context (see section.

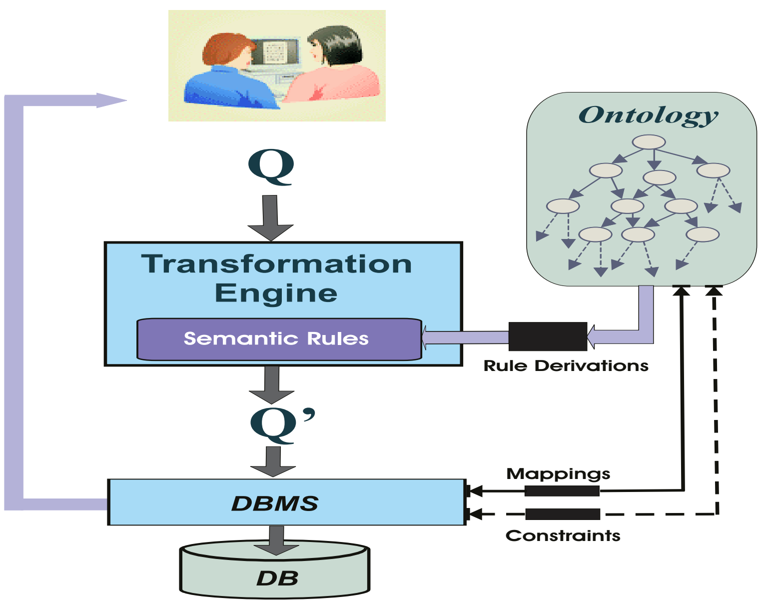

Fig. 2. System Architecture

During query reformulation semantic rules are applied uniformly, in any order. This is done iteratively such that at each iteration the reformulated query obtained in the previous iteration is used to generate another query until no more reformulations are possible. It is possible that no rules can be applied to the query and the output query is then equal to the input query. The rule derivation process is done manually by ontology and database experts. We have developed a set of such rules based on information mappings between ontological and database entities. The second component is an ontology which is associated with the underlying database. It could be either a general or a domain-oriented ontology depending on the nature of the database in question. Here, the role of the ontology is to provide semantic knowledge about the data in the database. Its content is adapted to the database instances in such way that it should be used correctly and completely (see section 5). The dashed arrow represents a set of constraints that must be satisfied for this purpose. The third component is the DBMS which processes the output query and returns the answer to the user. The answer might contain more or fewer tuples than that answer expected by the original query. According to this feature we classify the reformulation rules into two categories: *Augmentation rules* and *reduction rules*. In this paper we focus on the second class. For the first class we describe only one rule; please refer to [14] for additional rules.

Notations. Let U be a set of attributes $A_1, \ldots, A_n$ with domains $dom(A_i)$. Let DB be a database whose D is the set of all attribute domains. Let ID be the set of id-attributes of DB. The database schema is defined as a set of relation schemas R_1 ,. . .,R_m with $R_i \subseteq U$. We denote by $PKEYS(U)$, the set of primary Keys and by $FKEYS(R_i, R_j)$ the set of foreign keys in R_i to R_j. Furthermore, we choose the Domain relational calculus (DRC) to represent user queries [21]. Let δ_1 be the mapping that represents matchings between relation names of and ontology concepts called *relation-concepts*; δ_2 be the mapping that represents matchings between attribute names and ontology concepts called *attribute-concepts*, and δ_3 be the mapping that represents matchings between database values and ontology concepts called *value-concepts*. Finally, let δ_4 be the mapping that represents matchings between a pair of attribute names and ontology relationship-types.

4.1 Augmentation Rules

The goal of these rules is to extend the query answer with results that meet user's expectations. To this end, we have developed four rules: a Vocabulary-, a Support-, a Feature, and a Part-Whole rule [13, 14]. The first rule addresses semantic ambiguities discussed in section 2. The second rule is based on semantics of the relationships from which the ontology is constituted. The third rule is based on the domain-specific relationships that are mapped to the database model. In the following, we describe the fourth rule in details.

The basic idea of the Part-Whole rule is the use of the ”part-whole” properties to discover new database objects which are closely related to those the given query returns. Based on the semantic relationship `"PartOf"` the rule rewrites a user query by substituting the query terms by other semantically equivalent ones. For this rule, the concepts corresponding to the substituted terms together with the `"PartOf"`-relationships specify the same concepts corresponding to the original query terms. Thus, the same type of the object specified in the query can be defined in another way by using an alternative set of terms. A formal description of the rule is given in Appendix B.

For example, if a user wants to retrieve data about the Item ”pc” from the database DB_1, the query submitted may look like

$$Q_1 = \{(x_1, x_2, x_3, x_4)|\ (x_1, x_2, x_3, x_4) \in ITEM\ \wedge x_2 = "pc"\}.$$

This query asks for objects of type ”pc”. According to the ontology O_1 we deduce that a ”pc” is composed out of three parts: a ”desktop” , a ”monitor” and ” a ”keyboard”. Assuming that all PC-objects in the database are composed exactly out of these parts, which do not participate in the composition of any other object, enables the identification of PCs by means of their components. Thus, the set of terms {”desktop”, ”monitor”, ”keyboard”} and the term ”pc” are semantically equivalent.

By applying the Part-Whole rule to the query Q_1 we obtain a reformulated query $Q_1^{'}$ that retrieve also objects whose parts are the previous components. A formal description of $Q_1^{'}$ is given in Appendix B. Therefore, it is not surprising that the tuples 123 and 128 with attribute values ”computer” and ”product” meet fully the intention of the user. When a user poses the query Q_1 to the DB_1 database, these tuples will certainly be missed. As a result, the number of tuples will increase.

4.2 Reduction Rules

The main feature of these rules is that after reformulating a user query the number of tuples in the answer might decrease compared to that number of tuples before any reformulation.

In the following, we describe one of such rules. We call this rule *”the sensitivity rule”* because its goal is to increase the *sensitivity* of a user query. A query is called *sensitive* if its answer contains as few as possible *false positives*. We define a tuple as *false positive* if it is semantically not correct w.r.t. the user's expectations.

For example, a problem might occur when querying a database containing homonymous terms. If a user queries a database using terms in his query expression that might be homonymous with some other terms in that database, the answer to his query might contain tuples that are irrelevant to him. For instance, the term ”bank” has different meanings. It means either a container for keeping coins or a piece of furniture for sitting on or a financial institution for saving money [1]. Therefore, if a user queries a given database for information concerning an object ”bank”, the database might return tuples containing data about furniture, containers and institutions of type bank. This might not meet the user's intention if the user expects data only on furniture.

To solve this problem, we propose a reformulation rule based on the use of an ontology associated with the given database. By applying this rule a context could be specified for a user query. That is, the context defined by the semantic description of the data, which uses vocabularies from the ontology to express the user's intention. The intuition is to specify user queries sufficiently to derive the relevant meaning based on the ontology concepts. Thus, in the example above, the user's intention to find information about ”bank” as furniture can be specified by domain specific ontologies which can

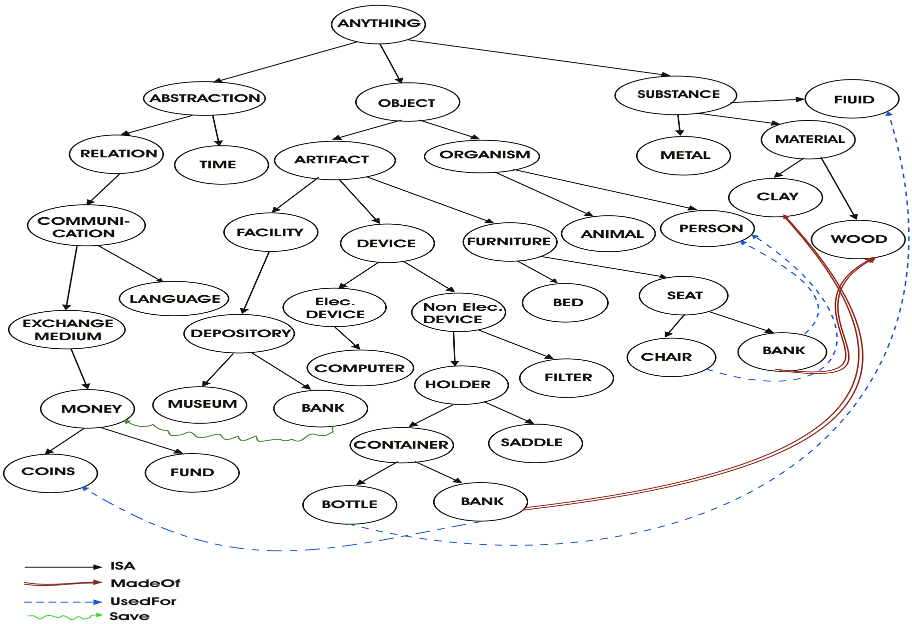

Fig. 3. Entity Ontology O_2

describe different aspects of furniture. Thus, the context of user queries is restricted to furniture. However, if the ontology is more general i.e. specifies more than one context (see Figure 3). In this case it would be difficult to determine the user's intention immediately. For example, the concept BANK might label two different nodes in two different subgraphs of the ontology. Each subgraph represents the related context of "bank". We suggest that the system asks the user to specify a unique "context". This could be done by providing him with the possibility to choose one of the ontology contexts in terms of the *immediate uncommon concepts* (imuc) of the BANK nodes. The immediate uncommon concepts of two given concepts are defined in terms of the *least common concept* (lcc) as follows:

Definition 1. Let a, b, l be concepts of ζ. l is a *least common concept* (lcc) of a and b iff

- $a \in DESC(\text{"}ISA\text{"}, l)$ and $b \in DESC(\text{"}ISA\text{"}, l)$,
- $\forall k, k' \in \zeta$, if $a, b \in DESC(\text{"}ISA\text{"}, k) \cap DESC(\text{"}ISA\text{"}, k')$ then $k = k'$
- if $\exists\, c' \in \zeta \mid a \in DESC(\text{"}ISA\text{"}, c')$ and $b \in DESC(\text{"}ISA\text{"}, c')$ then $l \in DESC(\text{"}ISA\text{"}, c')$ □

Definition 2. Let a, b, m, m', g, and g' be concepts of ζ. m and m' ($m \neq m'$) are *immediate uncommon concepts* (imuc) of a and b resp. iff

- $\exists\, l \in \zeta \mid l = lcc(a, b)$ AND
- $m = RChild(\text{"}ISA\text{"}, l) \wedge m' = RChild(\text{"}ISA\text{"}, l)$

For example, the immediate uncommon concepts of the concepts BOTTLE and CHAIR are the concepts DEVICE and FURNITURE, respectively, since their least common concept is the concept ARTIFACT. □

Next, we illustrate the sensitivity rule and its effectiveness by means of an example. A formal description of the rule is given in Appendix B.

Example 2. We assume a database DB_2 containing information about store items. The DB_2 schema might have a relation, called 'Store', whose schema defines the name of each object, the material it is made of, its use and its price. An instance of DB_2 and a description of the relation 'Store' are given as follows:

STORE(A-ID, Name, Made, Use, Price)
A-ID: Store identifier
Name: Store name
Made: Material type
Use: Purpose of use
Price: Item price
Primary-Key(A-ID)

In addition, we assume an ontology, denoted by O_2, which describes concepts of things. A portion of O_2 is adopted from [17, 6]. This ontology contains additional domain relationships: "MadeOf", "UseFor" and "Save". The meaning of "UseFor"-relationship, for example, is that if A (a concept) relates to B (a concept) by this relationship, the objects referred to A are used for purposes given by the objects referred to B. Figure 3 shows a graph representation of a portion of O_2. For the sake of clarity we omit some nodes and the other kinds of relationships.

Now, suppose that the user wants to retrieve all tuples from DB_2 concerning the container 'bank'. His query can be represented as following:

$$Q_2 = \{(x_1, x_2, x_3, x_4, x_5) \mid (x_1, x_2, x_3, x_4, x_5) \in STORE \wedge x_2 = \text{"}bank\text{"}\}.$$

Obviously, the answer from the current DB_2 database to the query Q_2 contains the tuples 42 and 47. However, the tuple 42 does not meet the intention of the user since it relates to furniture. By using the ontology O_2 the system could deduce that "bank" is related to three different contexts: Furniture, device and facility. This is done by retrieving the *imuc* of BANK concepts. Therefore, it has to ask again the user for specifying his query providing him the three relevant variants. If the user means a device "bank", the system will be able to specify the concept BANK from O_2 that the related objects are used for keeping coins. Thus, the user query should include terms represented by the concept COINS to assert the intended context of the answer. The application of the rule 2 to the query Q_2 leads to the following query:

$$Q_2^{'} = \{(x_1, x_2, x_3, x_4, x_5) \mid (x_1, x_2, x_3, x_4) \in STORE \wedge (x_2 = \text{"}bank\text{"} \wedge x_3 = \text{"}coins\text{"})\}.$$

The answer to this reformulated query will contain then only the tuple 47 as expected by the user. □

5 Semantic Model and Criteria

In this section, we propose a semantic model and two basic criteria which allow us to validate the reformulation rules and to ensure the consistency of the ontology with its underlying database. Due to the lack of space we will not describe the validation of the proposed rules, please refer to [14] for this issue.

5.1 Semantic Model

The semantic model is stated as an extension of the given ontology, denoted by O^*, which includes new concepts and additional relationship types. The new concepts represent relation names, attribute names and attribute values of the database unless they already exists. We denote these concepts by NC_{RN}, NC_{AN} and NC_V, respectively. We call *id-concepts* the concepts that represent id-values of the database and denote its set by Ω.

The additional relationships have to relate the new concepts to the existing ones or to each other. Their types are defined as follows:

– **"ValueOf"** is the type of relationship that relates each value-concept to its associated attribute-concept.
– **"HasA"** is the type of relationship between relation-concepts and attribute-concepts.
– **"InstanceOf"** is the type of relationship that relates an Id-concept to its associated relation-concept.
– **"TupleVal"** is the type of relationship that relates value-concepts to each other, which are associated with a particular tuple.

Figure 4 shows a portion of the semantic model O_1^* related to the ontology O_1 and the database DB_1.

In summary, the extended ontology is defined by $O^* = \{G^*, \zeta^*, \Re^*, \Im^*\}$ where $\zeta^* = \zeta \cup NC_V \cup NC_{AN} \cup NC_{RN}$, $\Re^* = \Re \cup \{\text{"ValueOf"}, \text{"InstanceOf"}, \text{"HasA"}$ $\text{"TupleVal"}\}$, and $\Im^*$ consists of all logical axioms related to $\Re^*$.

An extended ontology could also be expressed in a logical language. For instance, using the First Order Language (FOL) O^* can be defined as a theory Γ which consists of an Interpretation I and a set of well formed formulas [18]. I is specified by the set of individuals ζ^* and an interpretation function $\cdot^I$. Appendix C shows a logical interpretation of O_1^*.

5.2 Consistency Criteria

The basic consistency criteria are *correctness* and *completeness*, which aim at asserting the soundness of our framework. Hence, a set of constraints must be checked for applying correctly the transformation rules. These constraints affect the design of the ontology and the implementation of database instances. Note that the ontology must not be created from scratch but a preexisting one could be reused and adapted to the underlying database by respecting these constraints [20]. Similarly, database instances must satisfy the constraints specified by the ontology.

In order to formally define the consistency criteria we need the graph operator $SelectRel$ (see Appendix A). From a semantic point of view, if two id-concept nodes

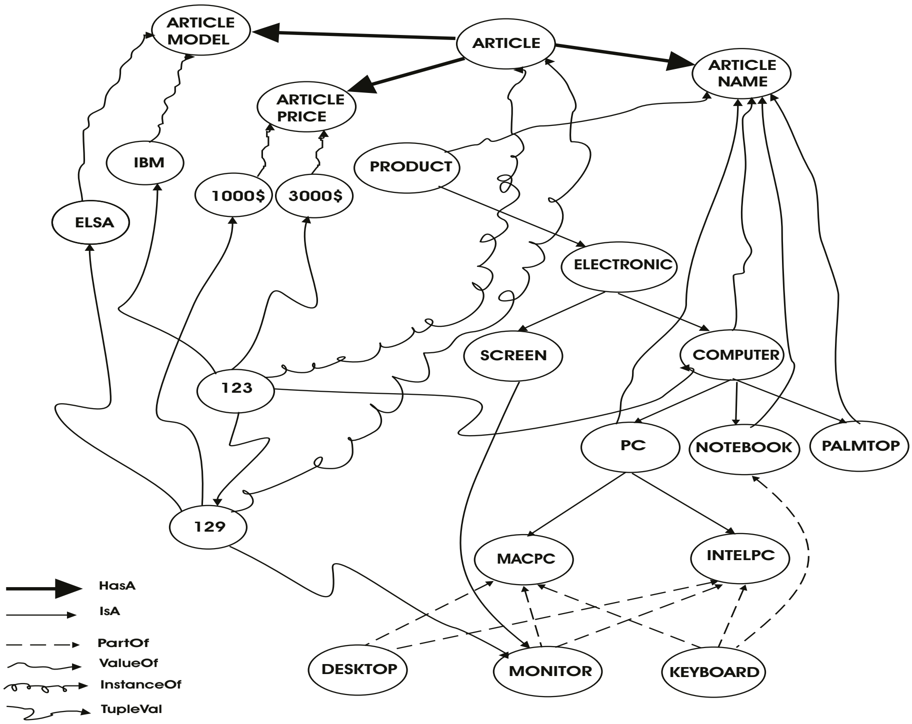

Fig. 4. A portion of the Semantic Model O_1^* in Figure 1

are adjacent (common edges are of type "`TupleVal`") then the semantic relationship between the represented concepts can be deduced from the result of the $SelectRel$ operation on these nodes. For example, if we apply $SelectRel$-operator on the concept nodes corresponding to the identifiers 123 and 129, we can deduce that the object identified by 129 is part of the object identified by 123. We denote by $|SelectRel_{PartOf}()|$ the number of "`PartOf`"-labels returned by the $SlectRel$-operator.

Definition 3. Let be $ic_1, ic_2 \in \Omega$. ic_1 and ic_2 are said to be *semantically dependent* if and only if $SelectRel(G^*, ic_1, ic_2) \neq \emptyset$.

Correctness Criterion. Intuitively, *correctness* means that any results of the reformulated query, say Q', can be ”derived” in the extended ontology O^* i.e. the concepts and relationships corresponding to database objects in the results of Q' must be correctly represented in the model O^*.
Formally, an extended ontology O^* is a correct model if and only if:
$\forall id_1, id_2 \in dom(ID)$, $R \in DB$, $ID \in PKEYS(R)$ such that $(id_1, c_1) \in \delta_3$ and $(id_2, c_2) \in \delta_3$:

1. IF $G^*(c_1, \text{"TupleVal"}, c_2)$ THEN c_1 and c_2 are semantically dependent
2. IF $|SelectRel_{PartOf}(G^*, c_1, c_2)| \neq \emptyset$ THEN $|SelectRel_{PartOf}(G^*, c_1, c_2)| = 1$

3. IF $\exists A_i, A_j \in R \mid (\{(A_i, A_j)\}, \beta_0) \in \delta_4$ THEN $\forall \mu \in R,\ \ G^*(c_i, \beta_0, c_j)$
where $\beta_0 \in \Re$, $i \neq j$, $(\mu[A_i], c_i) \in \delta_3$ and $(\mu[A_j], c_j) \in \delta_3$ □

The intuition behind the first constraint is that if two database tuples are semantically related, then there exist in O^* at least one semantic relationship between the two value-concepts associated with two attribute values of the tuples. The intuition behind the second constraint is that only a PartOf-relation level is allowed for all the database instances i.e. if item A is part of item B and item B is part of item C than the database does not store explicitly the relation: Item A is part of item C. The third constraint asserts that if a semantic relationship between two concepts representing two attribute names exists then the concepts representing the attribute values should be related to each other through the same relationship.

Completeness Criterion. Intuitively, *completeness* ensures that any tuple that is "derived" in O* for a given query Q' should also be in the answer of Q' i.e. the value-concepts together with their relationships corresponding to the results of Q' at the semantic level must be reflected in the database instance. Completeness constraints are formally described as follows:

1. $\forall id_1 a_v p \ \exists\ id_2\ Key(id_1) \wedge TUPVAL(id_1, a_v) \wedge PARTOF(a_v, p) \rightarrow$
$TUPVAL(id_1, id_2) \wedge TUPVAL(id_2, p) \wedge Key(id_2)$

2. $\forall id_1 a_v p\ \exists id_2\ Key(id_1) \wedge TUPVAL(id_1, a_v) \wedge COMMONPART(a_v, p) \rightarrow$
$Key(id_2) \wedge TUPVAL(id_1, id_2) \wedge TUPVAL(id_2, p)$ □

Due to limited space we describe the predicates of the above formulas in Appendix C. The first axiom asserts that each decomposition of a concept in the ontology must reflect the same decomposition for its associated value in the database instance. For example, each instance of the DB_1-database where the Item name is "pc" should have "desktop", "monitor" and "keyboard" instances. In addition, this condition asserts when the PartOf-relationship is transitive with respect to the ISA-relationship. A concept, say B, is a part of a concept, say A, if B is a part of all the sub-concepts of A. For example, the concept MONITOR is a part of the concept PC because it is a part of both concepts MACPC and INTELPC, which are sub-concepts of PC. On the other hand, the second axiom asserts that if all the sub-concepts of A (a concept) have a common part P (a concept) then each DB-instance reflecting A must be related to an instance which reflects P.

6 Mappings Between Ontologies and Databases

In order to accomplish the query reformulation task, mapping information between the ontology and the underlying database must exist. This information links the concepts and the relationships of the ontology with the database elements: Relations, attributes, attribute domains.

In this section, we focus on how to define these mappings rather than how to find them. Regarding the creation of mappings there is currently no automatic method for solving this issue but semi-automatic methods based on linguistic matchings might be adequate for this purpose [10]. In the following, we define the necessary properties that make such mappings adequate for applying the transformation rules. Moreover, we specify the necessary conditions for each kind of mappings that must be verified for maintaining the consistency between the ontology and its associated database. We address each aspect of the mappings separately.

6.1 Mapping Between Attribute Values and Concepts

We define a simple one-to-many mapping δ_3 for each value from the set of attribute domains D. The semantic of this mapping is that each value might be represented by a single or multiple concepts, but a given concept might represent at most one value. For example, the concept COMPUTER, in the O_2-ontology, is mapped to the value "computer" of the attribute Name in the relation Item. However, if a value has multiple homonyms, it might be represented by multiple concepts.
Formally, let A be an attribute name. We define δ_3 as a relation between ζ and D :

$\delta_3 \subseteq D \times \zeta$. Then, $\forall v_0 \in dom(A)\ \exists c_0 \mid (v_0, c_0) \in \delta_3$ and δ_3 is *injective*. □

In this context, each tuple in a given relation may be mapped to more than one concept. For example, tuple 43 in Table 3 can be mapped to two concepts related to the attribute values "chair" and "wood".

6.2 Mapping Between Attribute Names and the Ontology

Now, we define the mapping of attribute names to concepts and relationships of the ontology. Like the previous definitions, each attribute name might be mapped to one or more concepts in the ontology and each concept covers at most one attribute name. This mapping is also *injective*. In addition, if such mapping exists then the following constraints must be satisfied: Each value of the domain of that attribute must be mapped to a concept in the ontology. This concept must be related to the concept representing the attribute through the "ISA"-relationship.

Formally, let U be a set of attribute names. We define δ_2 as a relation between U and ζ: $\delta_2 \subseteq U \times \zeta$. Then, The following conditions must be satisfied:

Table 3. Store relation

A-ID	Name	Made	Use	Price
41	bed	wood	kid	120 $
42	bank	wood	person	300 $
43	chair	wood	person	150 $
44	flat iron	substance	clothes	60 $
45	chain	gold	women	850 $
46	perfume	roses	women	85 $
47	bank	clay	coins	50 $
48	cage	metal	birds	300 $

(i) IF $\delta_2(A) = c_0 \in \zeta$ THEN $\forall x \in dom(A), \exists\, c \in \zeta\,|(x, c) \in \delta_3$
and $c \in DESC(\text{"}\mathtt{ISA}\text{"}, c_0)$ (1) □

Furthermore, two attribute names, say A_1 and A_2, could be mapped to a single relationship-type in the ontology. The semantic of this mapping is that each concept corresponding to a value of A_1 must be related to a concept corresponding to a value of A_2 through this relationship.
Formally, we define δ_4 as a relation: $\delta_4 \subseteq (U \times U) \times \Re$. Then,

IF $(\{(A_1, A_2)\}, \beta_0) \in \delta_4$ THEN

(i) condition (1) holds for A_1 and A_2 and
(ii) $\forall x \in dom(A_1), \exists y \in dom(A_2) \mid \exists\, (c_x, \beta_0, c_y) \in G$
where $(x, c_x) \in \delta_3$, $(y, c_y) \in \delta_3$, and $\beta_0 \in \Re$. □

6.3 Mapping Between Relations and the Ontology

So far, we presented the mappings for attributes and attribute values. Now, we address the mapping from a given relation in the database to concepts and relationships in the ontology. Like previous mapping types, a relation name might be mapped to several concepts. This mapping is also *injective*. We define two kinds of mappings: *Complete* and *partial* mappings.
The mapping is called *partial* if there exists a single concept representing the relation name and at least one concept representing an attribute name of this relation. The latter concept must be related to the concept corresponding to the relation name through the `"ISA"`-relationship. On the other hand, the mapping is called *complete* if all attribute names of the relation (except the ID-attribute if it is generic) are represented in the ontology and satisfy the constraint above.
Formally, let R be a relation, U(R) be a set of its attributes. We define the mapping δ_3 as a relation between $\{R_1, \ldots R_n\}$ and ζ: $\delta_3 \subseteq \{R_1, \ldots R_n\} \times \zeta$.
Let $c_0 \in \zeta|\ (R, c_0) \in \delta_3$. Then, δ_3 is *complete* iff:

(i) $\forall A \in U(R) \mid\ \exists c\ \in \zeta\ , (A, c) \in \delta_2$ and $c \in DESC(\text{"}\mathtt{ISA}\text{"}, c_0)$. □

6.4 Additional Constraints for Mapping Attribute Values

In this section, we formulate a set of constraints to ensure that a semantic model O^* remains correct when introducing new concepts and relationships in the ontology O to represent database values which are not already represented in O.

So far, if O does not cover an attribute, say A i.e. there exists a set of attribute values of A, say V_0, which are not represented by concepts in O, then new concepts should be created in O. To this end, we propose the following principles: for each $v_0 \in V_0$,
- create a new node n_0 in G with label l: $(v_0, l) \in \delta_3$,
- if a node n exists that corresponds to A such that $(A, n) \in \delta_1$ then relate n_0 with that node using an edge of type `"ISA"`. Otherwise, relate it with the universal concept node using the same edge type. □

So far new concepts are introduced in O, relationships among them and between existing concepts should be determined. These relationships are specified using the map-

ping information defined between attribute pairs and ontological relationship-types. To this end, each tuple in the database, in which a value v_0 of V_0 appear is examined as follows:
Let μ be a tuple of a relation R, and $U(R)$= $\{A_0, A_1, \ldots, A_n\}$ so that $v_0 = \mu[A_0]$. If there exists $A_k \in U(R)$ such that $(\{(A_0, A_k)\}, \beta_0) \in \delta_4$, $\beta_0 \in \Re$, then:

- insert edges of type β_0 between the node corresponding to v_0 and the children nodes of the node corresponding to $\mu[A_k]$ (w.r.t. the ontology design choice).
- if the node corresponding to $\mu[A_k]$ has no children then insert one edge of type β_0 between the node corresponding to v_0 and that node corresponding to $\mu[A_k]$. □

Concerning the problem of homonyms, the intervention of an ontology expert is needed for this task.

7 Conclusion

Recently, there is a growing interest in ontologies for managing data in database and information systems. In fact, ontologies provide good supports for understanding the meaning of data. They are broadly used in information integration systems to overcome problems caused by the heterogeneity of data and to optimize query processing among the distributed sources. In this paper, we use ontologies within a single relational database and present an approach of query processing using semantic knowledge from a given ontology to reformulate a user query in such way that the query answer is meaningful to the users. To this end, we propose a set of query reformulation rules and illustrate their effectiveness by some running examples. Furthermore, we present a semantic model and two basic criteria to prove the soundness of our approach. We also illustrate the semantic of mappings between the ontology and the database.

In the future work, we intend to design and develop a prototype based on this approach. To this end, we attempt to reuse an existing ontology and adopt it with an associated database with respect to our framework. In addition, we intend to extend our approach to enable the use of the semantic rules in federated database systems.

References

[1] Online english dictionary. www.onelook.com, 2005.

[2] R. B. A. Borgida. Conceptual modeling with description logics. The Description Logic Handbook - Theory, Implementation and Applications. pp: 349-372. Cambridge University Press, 2003.

[3] S. Bergamaschi, C. Sartori, D. Beneventano, and M. Vincini. ODB-tools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. Advances in Artificial Intelligence, 5th Congress of the Italian Association for Artificial Intelligence, Rome, Italy, 1997.

[4] B. Chandrasekaran, J. Josephson, and V. Benjamins. What are ontologies, and why do we need them? In *IEEE Intelligent Systems*, pages 20–26, 1999.

[5] M. Dzbor, J. Domingue, and E. Motta. Magpie- towards a semantic web browser. In *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 690–705, 2003.

[6] C. Fellbaum. Wordnet an electronic lexical database, 1998.

[7] T. Gruber. A translation approach to portable ontology specifications. In *Knowledge Acquisition (5) No. 2, USA*, pages 199–220, 1993.

[8] N. Guarino and P. Giaretta. Ontologies and knowledge bases: towards a terminological clarification. In *Knowledge Building Knowledge Sharing,ION Press*, pages 25–32, 1995.

[9] J. Heflin and J. Hendler. Dynamic ontologies on the web. In *AAAI/IAAI*, pages 443–449, 2000.

[10] M. Hernandez, R. J. Miller, and L. M. Haas. Clio: A semi-automatic tool for schema mapping. In *ACM SIGMOD*, 2001.

[11] A. Kayed and R. Colomb. Extracting ontological concepts for tendering conceptual structures. *Data and Knowledge Engineering*, 41(1-4), 2001.

[12] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Conference on Cooperative Information Systems*, 41:14–25, 1996.

[13] C. B. Necib and J. Freytag. Ontology based Query Processing in Database Management Systems. In *Proceeding on the 6 th international conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE'2003)*, pages 37–99, 2003.

[14] C. B. Necib and J. Freytag. Using Ontologies for Database Query Reformulation. In *Proceeding on the 18 th conference on Advances in Databases and Information Systems (ADBIS'2004)*, 2004.

[15] N. Noy and C. D. Hafner. The state of the art in ontology design. *AI Magazine*, 3(18):53–74, 1997.

[16] N. Paton, R. Stevens, P. Baker, C. Goble, S. Bechhofer, and A. Brass. Query processing in the TAMBIS bioinformatics source integration system. *Statistical and Scientific Database Management*, pages 138–147, 1999.

[17] A. Pease. The sigma ontology development environment. In *IJCAI-03 Workshop on Ontologies and Distributed Systems (ODS'03),Acapulco, Mexico*, Lecture Notes in Computer Science, 2003.

[18] M. Peim, E. Franconi, N. Paton, and C. Goble. Query processing with description logic ontologies over object-wrapped databases. technical report, University of Manchester, 2001.

[19] A. Perez and V. Benjamins. Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, 1999.

[20] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1-2):161–197, 1998.

[21] J. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.

[22] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information- a survey of existing approaches. In *Proc. of IJCAI*, 2001.

A Graphical Operations

A set of primitive graph operations: $ISAChild$, $RChild$, $RParent$, $ANCES$, $DESC$, $SUBT$, $SYNs$ and $PARTs$, are needed for formal representations of the transformation rules.

Let $P_{ths}(c_i - c_j)$ be a set of directed paths between two concept nodes c_i and c_j. Let be $c_1, c_2, s_k, s_h \in \zeta$ and $R, R_i \in \Re$:

- $ISAChild(c_1) = \{c_2 \mid G(c_1, \text{"}\mathtt{ISA}\text{"}, c_2)\}$
- $SYNs(c_1) = \{c_2 \mid G(c_1, \text{"}\mathtt{SynOf}\text{"}, c_2)\}$
- $Rchild(R, c_1) = \{c_2 \mid G(c_1, R, c_2)\}$
- $Rparent(R, c_1) = \{c_2 \mid G(c_2, R, c_1)\}$
- $SUBT(c_1) = \{c_2 \in \zeta \mid \exists P_{ths}(c_1 - c_2)\}$
- $DESC(R, c_1) = \{c_2 \in \zeta \mid \exists p \in P_{ths}(c_1 - c_2) : \forall e = (s_k R_i s_h) \in p\ ,\ R_i = R\}$
- $ANCES(R, c_1) = \{c_2 \in \zeta \mid \exists p \in P_{ths}(c_2 - c_1) : \forall e = (s_k R_i s_h) \in p\ ,\ R_i = R\}$
- $SelectRel(G^*, c_1, c_2) = \{R_i \in \Re \mid \exists A, B \in V, \exists\ P \in P_{ths}(A, B) :$
  $G^*(c_1, \text{"}\mathtt{TupleVal}\text{"}, \mathtt{A}), \mathtt{G}^*(\mathtt{c}_2, \text{"}\mathtt{TupleVal}\text{"}, \mathtt{B}) \wedge \exists\ \mathtt{s_k R_i s_h} \in \mathtt{P}\}$ □

Informally, $ISAChild(c)$ is the set of the immediate sub-concepts of c (a concept). $Rchild(R, c)$ is the set of all descendant concepts of c following edges of type R. Similarly, $Rparent(R, c)$ is the set of all ascendant concepts of c following edges of type R. $DESC(R, c)$ returns the set of all descendant concepts of c following edges of type R, whereas $ANCES(R, c)$ returns the set of all ascendant concepts of c by following edges of type r. Similarly, $SUBT(c)$ returns all descendants of c for any edge-type and $SYNs(c)$ returns the set of all synonyms of c. *SelectRel* returns all edge types of the paths between two concepts connected with other concepts via edges of type `"TupleVal"`.

In addition, we define an $Outgoings(c)$ as a set of edge-types going out from the node of a concept c. We also define a $PARTs(c)$ as a set of concepts that are "parts" of the concept c. According to our ontology graph design $PARTs(c)$ is determined by traversing the nodes related with to c following only edges of type "`PartOf`" and "`ISA`". More precisely, two cases must be distinguished:

- Case 1: If $Outgoings(c) \ni \text{"}\mathtt{PartOf}\text{"}$ then $PARTs(c) = A \cup B \cup C$ where
  - $A = DESC(\text{"}\mathtt{PartOf}\text{"}, \mathtt{c})$
  - $B = DESC(\text{"}\mathtt{ISA}\text{"}, \mathtt{a})$, $a \in A$
  - $C = SYNs(h) \cup SYNs(l)$, $h \in A$ and $l \in B$.

  Informally, $PARTs(c)$ is the set of concepts obtained by retrieving the labels of all nodes that are `PartOf`-children of the node c together with their `ISA`-descendants and synonyms.
- Case 2: If $Outgoings(c) \ni \text{"}\mathtt{ISA}\text{"}$ then $PARTs(c) = PARTs(s_i)$
  where $s_i \in A$ and $\forall (s_1, s_2) \in A^2\ PARTs(s_1) = PARTs(s_2)$,
  $A = DESC(\text{"}\mathtt{ISA}\text{"}, \mathtt{c})$.

  Informally, $PARTs$ of a concept c is defined recursively in terms of its sub-concepts. It is equal to the $PARTs$ of one of its sub-concepts (if they have the same $PARTs$). □

B Syntax of Query Reformulation Rules

Let be $t_0 \in D$ and $R, R_1, R_2 \in DB$. Part-Whole and Sensitivity rules are formulated as follows.

B.1 Part-Whole Rule

IF $Q = \{(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in R_1 \wedge x_i \theta t_0\}$
and $\exists A_1, A_2 \in FKEYS(R_2, R_1) | \ \delta_4(A_1, A_2) = \text{"}PartOf\text{"}$
and $\exists\, c_0 \in \ \zeta \mid \delta_3(t_0) = c_0$
and $\forall\, c_i \in \zeta,\ c_i \neq c_0,\ PARTs(c_0) \ \not\subseteq PARTs(c_i)$

THEN $Q^{'} = \{(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in R_1 \wedge x_i \theta t_0\} \ \cup$
$\{(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in R_1 \wedge \ [\exists (y_{11}, \ldots, y_{n1}) | (y_{11}, \ldots, y_{n1}) \in R_2 \wedge$
$x_1 = y_{11} \wedge \exists (z_{11}, \ldots, z_{n1}) \in R_1 \wedge (z_{11} = y_{21} \wedge z_{i1} = s_1)] \ \wedge \ldots \wedge [\exists (y_{1m}, \ldots, y_{nm)} |$
$(y_{1m}, \ldots, y_{nm}) \in R_2 \wedge x_1 = y_{1m} \wedge \ \exists (z_{1m}, \ldots, z_{nm}) \in R_1 \wedge (z_{1m} = y_{2m} \wedge z_{im} =$
$s_m)]\}$
where $s_j \in I_0 = \{t \in D \,|\, \delta_3(t) \in PARTs(c_0)\}$, $1 =< j =< m = |I_0|$.

By applying this rule on the query Q_1 (section r̃efaugmentation rules), the reformulated query is given as follows:

$Q_1^{'} = \{(a_1, a_2, a_3, a_4) \mid (a_1, a_2, a_3, a_4) \in ITEM \ \wedge a_2 = \text{"}pc\text{"}\} \ \cup$
$\{(a_1, a_2, a_3, a_4) \mid (a_1, a_2, a_3, a_4) \in ITEM \wedge [\exists\ y_1, y_2 | (y_1, y_2) \in COMPONENT$
$\wedge a_1 = y_1 \wedge \exists (b_1, b_2, b_3, b_4) \in ITEM \wedge (y_2 = b_1 \wedge b_2 = \text{"}monitor\text{"})] \wedge$
$[\exists\ z_1, z_2 | (z_1, z_2) \in COMPONENT \wedge a_1 = z_1 \wedge \exists (c_1, c_2, c_3, c_4) \in ITEM \wedge (z_2 =$
$c_1 \wedge c_2 = \text{"}keyboard\text{"})] \wedge [\exists\ u_1, u_2 | (u_1, u_2) \in COMPONENT \wedge$
$a_1 = u_1 \wedge [\exists\ d_1, d_2, d_3, d_4 | (d_1, d_2, d_3, d_4) \in ITEM \wedge u_2 = d_1 \wedge d_2 = \text{"}desktop\text{"})]\}$

B.2 Sensitivity Rule

IF $Q = \{x_i \mid (x_1, \ldots, x_n) \in R \ \wedge \ x_p \theta t_0\}$

and $\exists\, c_0, c_p \ \in \zeta \,|\, \delta_3(t_0) = c_0$ and $c_p = \delta_2(A_p)$
and $\exists A_i, \ldots, A_j \in U(R) \mid \delta_4(A_p, A_k) = r_k \in \Re \setminus \{\text{"}PartOf\text{"}\}$
and $c_0 \in SUBT(c_k) \cap SUBT(c_p),\ c_k = \delta_2(A_k)$

THEN $Q^{'} = \{(x_1, \ldots, x_n) | \ (x_1, \ldots, x_n) \in R \wedge (x_p \theta t_0) \bigwedge_{k=i}^{j} (\bigvee_{h=1}^{m} x_k \theta t_{kh})\}$

where $t_{kh} \in I_0 \cup I_1 \cup I_2$, $m = |I_0 \cup I_1 \cup I_2|$
$I_0 = \{t \in D \mid \delta_3(t) = c_{kh}\}$
$I_1 = \{t \in D \mid \delta_3(t) \in DESC(\text{"ISA"}, c_{kh})\}$
$I_2 = \{t \in D | \delta_3(t) \in SYNs(c_{kh}) \vee \in SYNs(a),\ a \in DESC(\text{"ISA"}, c_{kh})\}$
$c_{kh} \in DESC(\text{"}ISA\text{"}, c_k) \cap RParent(r_k, c_0)$, and
$i =< k =< j =< n, k \neq p,\ 1 =< h =< m.$ □

C Logical Interpretation

Γ: $I = (\zeta^*, \cdot^I)$
$ISA^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{ISA}\text{"}, b)\}$
$SYN^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{SynOf}\text{"}, b)\}$
$PARTOF^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{PartOf}\text{"}, b)\}$
$HASA^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{HasA}\text{"}, b)\}$
$VALUEOF^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{ValueOf}\text{"}, b)\}$
$INSOF^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{InstanceOf}\text{"}, b)\}$
$Key^I = \{a \in \zeta^* | \; G^*(a, \text{"}\texttt{InstanceOf}\text{"}, b)\}$
$TUPVAL^I = \{(a,b) \in \zeta^{*2} | \; G^*(a, \text{"}\texttt{TupleVal}\text{"}, b)\}$
$WHOLE^I = \{a \in \zeta^* | \forall \; b_1 \; b_2 \; c \; ISA(a, b_1) \wedge ISA(a, b_2) \wedge PARTOF(b_1, c) \rightarrow PARTOF(b_2, c)\}$

$\forall x. \; ISA(x, x)$
$\forall x. \; SYN(x, x)$
$\forall x. \; PARTOF(x, x)$
$\forall xyz. \; ISA(x, y) \wedge ISA(y, z) \rightarrow ISA(x, z)$
$\forall x.y \; SYN(x, y) \leftrightarrow SYN(y, x)$
$\forall xyz. \; SYN(x, y) \wedge SYN(y, z) \rightarrow SYN(x, z)$
$\forall xyz. \; ISA(x, y) \wedge SYN(y, z) \leftrightarrow ISA(x, z)$
$\forall xyz. \; ISA(x, z) \wedge SYN(x, y) \leftrightarrow ISA(y, z)$
$\forall xy \; \exists \; z. \; VALUEOF(x, y) \rightarrow HASA(z, y)$
$\forall xy \exists z. TUPVAL(x, y) \rightarrow INSOF(x, z)$
$\forall xyz. \; PARTOF(x, y) \wedge SYN(y, z) \leftrightarrow PARTOF(x, z)$
$\forall xyz. \; PARTOF(x, y) \wedge SYN(x, z) \leftrightarrow PARTOF(z, y)$
$\forall xyz. \; PARTOF(x, y) \wedge PARTOF(y, z) \rightarrow PARTOF(x, z)$
$\forall xyz. \; VALUEOF(y, z) \wedge ISA(x, y) \rightarrow VALUEOF(x, z)$
$\forall xyz. \; VALUEOF(y, z) \wedge SYN(x, y) \rightarrow VALUEOF(x, z)$
$\forall xyz. \; \exists \; w. \; INSOF(x, y) \wedge HASA(y, z) \rightarrow TUPVAL(x, w) \wedge VALUEOF(w, z)$
$\forall xyz. \; WHOLE(x) \wedge ISA(x, y) \wedge PARTOF(y, z) \leftrightarrow PARTOF(x, z)$
$\forall xyz_1z_2. \; COMMONPART(x, y) \leftrightarrow ISA(x, z_1) \wedge ISA(x, z_2) \wedge PARTOF(z_1, y) \wedge PARTOF(z_2, y)$

x, y, w, z, z_1, z_2 are variables. □