A Case-Based Approach to Information Integration

Maurizio Panti[a] Luca Spalazzi[a] Alberto Giretti[b]

[a]Istituto di Informatica, University of Ancona, via Brecce Bianche, 60131 Ancona, Italy
[b]IDAU, University of Ancona, via Brecce Bianche, 60131 Ancona, Italy
{panti,spalazzi}@inform.unian.it giretti@idau.unian.it

Abstract

Information integration is the problem of taking information from distributed, heterogeneous, and often dynamic sources and making them work together as a whole. A number of ideas concerning information integration are based on the notion of rewriting queries. In this paper we propose a distributed case-based approach to the problem of rewriting queries. According to this approach we use a case memory instead of static views, i.e. views that are defined a priori. As a consequence, the mediated schema is dynamically updated, strongly influenced by the queries submitted by a consumer. This approach allows a mediator to face systems where consumers may change their customization needs and information sources may become unavailable, may be added, or may modify their schemas.

1 Introduction

Nowadays, information systems can be thought as collections of *information producers* (i.e., information sources) and *information consumers* (i.e., users that perform transactions) that are often *distributed* in world-wide networks. This means that producers and consumers can be autonomous and, as a consequence, are often *heterogenous* and *dynamic*. Information sources are heterogenous due to discrepancies at the physical level (different DBMSs, legacy applications, software and hardware platforms), logical level (different data models), and conceptual level (different schemas, concept and relation names). Moreover autonomous information sources are dynamic. Basically this is due to the fact that they may be added to the system, or become (temporarily or definitively) unavailable. Sometimes, they may also vary their conceptual schemas. Information consumers are heterogeneous and dynamic as well. Indeed new consumers can be inserted or removed. Furthermore, each consumer may have different customization needs due to distinct business objectives and these needs can change very often. In this work, we focus on information integration in distributed, heterogeneous, and dynamic information systems, i.e., constructing answers to query from consumers. The problem consists on rewriting a consumer's query into queries to specific information sources. In this paper we present a distributed case-based approach to the problem of rewriting queries. According to this approach we have a dynamic mediated schema instead of a static one, i.e., a schema that is defined a priori.

Proceedings of the 26th VLDB Conference, Cairo, Egypt, 2000.

2 Related Work

Up to now, several approaches have been proposed for information integration in distributed, heterogeneous, and dynamic information systems.

Classic Approach [30, 32, 1, 23]. This approach relies on building a single global schema to encompass the differences among multiple local source schemas. The mapping from the global schema to each local schema is often expressed in a common SQL-like language (e.g., HOSQL [1] and SQL/M [23]). The enforcement of a single global schema through data integration yields full transparency for uniform access to distributed and heterogeneous information sources. Nevertheless this approach does not fit well in the case of dynamic producers since the global schema is statically built a priori. Moreover, a single global schema does not support different needs of heterogeneous and

dynamic consumers.

Federated Approach [33]. This approach improves the previous approach dealing with consumer heterogeneity. Indeed it relies on multiple integrated schemas. However, the heterogeneity problems are resolved at the schema integration stage and integrated schemas are static. This approach cannot scale well when new sources or consumers need to be added or removed. Moreover, source schemas or consumer requirements cannot be upgraded without a revision of the integrated schemas.

Distributed Object Management Approach [27, 28, 8]. This approach generalizes the federated approach by modeling distributed and heterogeneous databases as collections of objects in a distributed object space. It is based on a common object model and a common object query language (e.g., the ODMG standard [8]). Nevertheless, this approach does not yield full transparency for uniform access to sources that are heterogeneous at the conceptual level.

Intelligent Information Integration (I^3) Approach [35, 19]. This approach relies on the so called *mediator architecture*: a three-layers system architecture for information integration. A layer is devoted to information consumers, another layer is devoted to information producers, the middle layer deals with mediation, i.e., the original query is reformulated in a set of queries, each targeted at a selected source. There are two basic approaches to intelligent information integration: the procedural approach and the declarative approach. In the *procedural* approach (e.g., TSIMMIS [19], Squirrel [37], and WHIPS [21]), mediators integrate information from sources through ad-hoc procedures defined with respect to a set of predefined information needs. When such needs or sources change (i.e., we have a dynamic information system), a new mediator must be generated. In the *declarative* approach (e.g., Carnot [11], SIMS [2], Information Manifold [24], Infomaster [16], and CDLNR [7]), mediators use suitable mechanisms to rewrite queries according to source descriptions. Intuitively, a rewritten query would be equivalent to the original query (i.e., denote the same set of instances). Nevertheless, often this is not possible. As a consequence, "in information-integration applications, [query] containment appears to be more fundamental than equivalence" [34]. Query containment has been related to information integration via an approach called *synthesizing queries from views* [36, 9, 34, 3, 17]. According to this approach, a query reformulation problem becomes the problem of finding a solution (in terms of views) that must be contained in the original query.

Related to the declarative approach is the use of *description logics* [5, 4, 6] (see Appendix A) as a data modeling language and as a query language. This is the approach followed by several authors, see for example [2, 3, 7, 24, 34]. Indeed, description logics offer an interesting tradeoff between complexity and expressive power. In this perspective, it is worth to notice that the problem of query containment corresponds to the subsumption problem. Subsumption is a tractable problem for most of the description logics [6, 4] while query containment without any restriction is undecidable [3].

Among declarative based I^3 projects, only a few have dealt with dynamic information producers and consumers. For example Information Manifold [24], SIMS [2], and Infomaster [16] do not allow automatic adaptation of source descriptions even if the authors claim that a new (view of a) source can be easily added by writing its description without redefining the mediator. For example, this is the case when a consumer's need change and thus it cannot be satisfied by the given views.

Case-Based Approach. Case-based reasoning is a problem solving methodology which is based on previously experienced, concrete problem situations [26]. According to this approach, a system learns by experience how problems can be solved, therefore it is appropriate for dynamic application domains, when it is impossible to have predefined solution. Nevertheless, as far as we know, its application to information integration is a novel approach. The closest applications are in information retrieval [13, 31] (i.e., the extraction of information from non-structured data) and associative query answering [18, 10] (i.e., associating relevant additional information to a query answer). Even if these systems deal with queries, usually they do not deal with distributed information sources. A noteworthy example of distributed computing by means of case-based reasoning is the so called *distributed case-based reasoning* (DCBR). In DCBR there are several agents; each agent has its own case memory since it could have acquired its own independent problem-solving experiences. A new problem is solved through agent cooperation. Indeed each agent reuses the local past case that best contributes to the overall case. For example, CBR-TEAM [29] has a negotiation-driven case retrieval algorithm applied to distributed mechanical design problems.

Description logic is applied to case-based reasoning as well [25, 22, 12, 20]. Indeed, in case-based reasoning, subsumption becomes a powerful tool to automatically derive case hierarchies that can be used in case retrieval and case retention.

3 Work Overview

Our goal is to build an information system capable of interconnecting information consumers and producers. Previous approaches solve several problems concerned with distributed, heterogeneous information systems. Their main limitation is related to the capability of evolving according to dynamic information systems. We propose a system (see also [15]) which is based on

a mediator architecture in order to support customizable information integration across distributed, heterogeneous, and dynamic information sources. Source heterogeneity at the physical level is removed by suitable **wrappers**. Indeed each information source has its own management system and query language, it can use a very recent technology or be a legacy system. As a consequence, it is needed a sort of interface (a wrapper) between a source and the rest of the system. Wrappers translate queries in the local format and answers from the local format in the common language. Source heterogeneity at logical and conceptual levels is removed by **mediators** and their mediated schemas. Consumer heterogeneity is removed by the presence of several mediators since each of them is related to a consumer (or a class of consumers). Finally, the effects of dynamic sources and consumers are removed by mediators since they are able to dynamically update their schemas. Indeed when a mediator does not have enough local knowledge to reformulate a query, it can cooperate with sources (to access their original schemas) and other mediators (to access their mediated schemas).

Mediators capability to face heterogeneous and dynamic systems relies on a thesaurus and a distributed case-based reasoning. Each mediator has its own *thesaurus* in order to solve name heterogeneity. A thesaurus is composed by a set of classes of synonyms. Each element of a class has the reference to the information sources that use it as term. Every time a new query arrives, its terms are translated in standard terms by means of the thesaurus. The reverse process occurs when the rewritten query must be sent to information sources. The thesaurus must be dynamically updated when a change in the system occurs. The classes of the thesaurus are modified by means of clustering. The explanation of this technique is out of the scope of the paper, for more details we remind to [14]. Each mediator has also its own *case-based reasoner* in order to solve heterogeneity of schemas and consumer's needs. Each case contains a query and its reformulation, and it is stored in the mediator's case memory. When a new query from the consumer arrives, the mediator looks for a past query similar to the new one and adapts the corresponding solution in order to obtain a reformulation of the new query. This means that the mediated schema of a mediator is strongly influenced by the queries submitted by consumers (mediator's experience). When this experience does not help, the mediator interacts with other mediators and/or information sources. This approach allows the mediator to update its schemas and therefore to take into account changes in information sources and consumer's needs. This is called distributed case-based reasoning.

The rest of the paper is organized as follows. A running example is introduced in Section 4. Section 5 describes how cases are represented. Local and distributed query rewriting are discussed in sections 6 and 7, respectively. Finally, some conclusions are given in Section 8.

4 A Running Example

We consider an information system which contains computer science bibliography data, namely information about articles and their authors. This information system is composed by several sources, consumers, and mediators. Figure 1 depicts three information sources of the system. Since the description of the thesaurus is out of scope of this paper, we consider that all the (concept and role) names have been already translated and thus no name heterogeneity occurs. Nevertheless, their conceptual schemas are different. The source w_1 (Figure 1.a) contains a set of authors and a set of articles classified according their publication type, i.e., *journal*, *conference*, etc. The source w_2 (Figure 1.b) contains a set of authors and a set of articles on database classified according their topic, i.e., *object_oriented* databases, *active* databases, *federated* databases, etc. Finally, the source w_3 (Figure 1.c) contains a set of authors and a set of articles on artificial intelligence classified according their topic as well, i.e., *planning*, *cbr*, *agents*, etc.

5 Query Representation

Generally speaking, a case is an arbitrary set of features (attribute-value pairs). Some features are devoted to represent a problem (in our application the query to be reformulated) in order to make easier the retrieval of a past problem similar to the current situation. The rest of features are devoted to represent the problem solution (the reformulated query and information sources where the reformulated query has been sent) to reuse it in the current problem. Furthermore, the problem of rewriting queries has a fundamental condition that must be satisfied: *the rewritten query must be contained in the original query* [3, 34]. We use a description logic as a language for representing queries and thus cases. The subsumption relation is used as query containment and thus for case retrieval. According to above considerations we define a case as follows:

Definition 5.1 (Case) *Let $w_1, \ldots, w_n$ be information sources. Let $Q_1, \ldots, Q_n$ be queries to $w_1, \ldots, w_n$ respectively. Let $Sol(Q)$ be a query obtained by an arbitrary combination of $Q_1, \ldots, Q_n$. Let Q be a query such that*

$$Sol(Q) \sqsubseteq Q \tag{1}$$

Then $Sol(Q)$ is a rewriting of the query Q and $\langle Q, Sol(Q), \langle (Q_1, w_1), \ldots (Q_n, w_n) \rangle \rangle$ is a case.

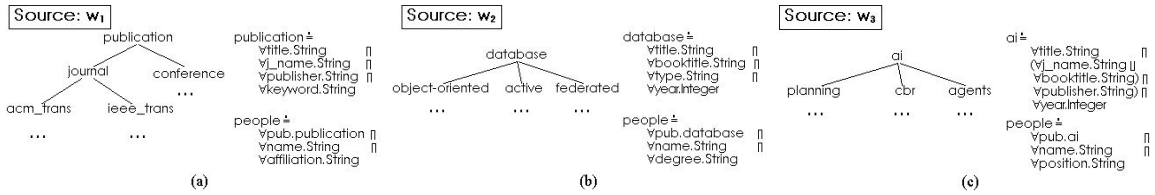

Figure 1: An example of information sources.

A collection of cases is called **case memory**. A **terminology** (i.e., a set of concept definitions and their relations) is always associated to a case memory. $\top, \bot$ represent the boundaries of the terminology w.r.t. the subsumption relation.

Example 5.1 In the example presented in Section 4 let us consider a mediator that has the following cases:

$$
\begin{array}{l}
\ldots\ldots \\
\langle H, (\forall pub.conference \sqcap \forall pub.db) \sqcup \forall pub.cbr, \\
\quad \langle (\forall pub.conference, w_1), (\forall pub.db, w_2), (\forall pub.cbr, w_3) \rangle \ \rangle \\
\langle I, \ \forall pub.acm\_trans \sqcup \\
\quad \forall pub.(ai \sqcap \forall j\_name.String \sqcap \\
\qquad \forall publisher.\{\text{“}ACM\text{”}\}) \\
\quad \langle (\forall pub.acm\_trans, w_1), \\
\quad (\forall pub.(ai \sqcap \forall j\_name.String \sqcap \\
\qquad \forall publisher.\{\text{“}ACM\text{”}\}), w_3) \rangle \qquad \rangle \\
\langle J, \ \forall pub.journal \sqcup \forall pub.ai, \\
\quad \langle (\forall pub.journal, w_1), (\forall pub.ai, w_3) \rangle \qquad \rangle \\
\langle K, \forall pub.\exists keyword.\{\text{“}Agents\text{”}\} \sqcup \forall pub.agents, \\
\quad \langle (\forall pub.\exists keyword.\{\text{“}Agents\text{”}\}, w_1), (\forall pub.agents, w_3) \rangle \ \rangle \\
\quad \ldots\ldots
\end{array}
$$

and the corresponding terminology:

$$
\begin{array}{l}
\{\ldots journal \dot{\leq} \top \ , \ acm\_trans \dot{\leq} journal \ , \\
\quad acm\_trans \doteq \forall publisher.\{\text{“}ACM\text{”}\} \ , \\
\quad ai \dot{\leq} \top \ , \ agents \dot{\leq} ai \ , \ cbr \dot{\leq} ai \ , \ db \dot{\leq} \top \ , \\
\quad A \doteq \neg journal \sqcap db \ , \ H \doteq \forall pub.(A \sqcup cbr) \ , \\
\quad I \doteq \forall pub.acm\_trans \ , \ J \doteq \forall pub.(journal \sqcup ai) \ , \\
\quad K \doteq \forall pub.agents \qquad \ldots\}
\end{array}
$$

A pictorial view of the terminology is given in Figure 2. For example the evaluation of $Sol(H)$ consists on querying sources w_1, w_2, and w_3 with $\forall pub.conference$, $\forall pub.db$, and $\forall pub.cbr$ respectively, and integrating the related answers. The semantics of concepts helps us on integrating the answers. Let $I_{w_1}(\forall pub.conference)$, $I_{w_2}(\forall pub.db)$, and $I_{w_3}(\forall pub.cbr)$ be answers from w_1, w_2, and w_3 respectively, then the answer to $\forall pub.(A \sqcup cbr)$ is: $(I_{w_1}(\forall pub.conference) \cap I_{w_2}(\forall pub.db)) \cup I_{w_3}(\forall pub.cbr)$. Notice that this is just one possible answer, another mediator with a different case memory may rewrite the query in a different way and thus return a different answer.

6 Local Query Retrieval and Reuse

Every time a new query arrives, it is inserted in the terminology and classified by means of subsumption. If there exists a past problem equal to the new query, then its solution can be used as solution for the new query. Otherwise, the solutions of the past problems that are subsumed by the new query can be used as solution for the new problem. The closest to the new query the retrieved cases are, the best the solutions are. This drives us to the notion of problems maximally contained in the new query [3, 34]. This notion allows the definition of the retrieval function $M_{inf}(Q, \mathcal{P})$ as a set of conjunctions of concepts $(\sqcap_{i \geq 1} X_i)$ of the given terminology $(\mathcal{P})$. For each conjunction of concepts there does not exist another conjunction of concepts of the same terminology $(\sqcap_{j \geq 1} Y_j)$ such that $\sqcap_{i \geq 1} X_i \sqsubseteq \sqcap_{j \geq 1} Y_j \sqsubseteq Q$.

Example 6.1 Let us suppose to have $Q \doteq \forall pub.ai$ as input query in Example 5.1. Applying the retrieval function we obtain: $M_{inf}(Q, \mathcal{P}) = \{K, H \sqcap J\}$

Now, we can obtain two different reuse methods depending on whether or not the original query is decomposed before the application of the retrieval methods. In the first approach, the system retrieves a solution if and only if the new query subsumes some past problems, if so past solutions are simply replied. The algorithm is reported below.

$$
\mathcal{S}_{inf}(Q, \mathcal{P}) \doteq \begin{cases} \bigsqcup_{\sqcap_i X_i \in M_{inf}(Q,\mathcal{P})} \sqcap_i Sol(X_i) \\ \qquad \text{if } M_{inf}(Q, \mathcal{P}) \neq \emptyset \\ \bot \qquad \text{if } M_{inf}(Q, \mathcal{P}) = \emptyset \end{cases} \tag{2}
$$

Example 6.2 Applying the basic reuse algorithms to case memory and input case of Example 6.1 we obtain: $\mathcal{S}_{inf}(Q, \mathcal{P}) = Sol(K) \sqcup (Sol(H) \sqcap Sol(J))$

The second approach is based on the combination of past solutions according to the new query structure. It is based on a decomposition algorithm that exploits the algorithm 2. The resulting algorithm is described in Figure 3. Notice that $Dual\_\mathcal{D}$ is the dual algorithm of $\mathcal{D}$. It retrieves concepts that minimally contain the given query. Moreover, when an unsuccessful termination occurs, the algorithm backtracks and stops the query decomposition to a higher level.

Example 6.3 Let us suppose to have $Q' \doteq \forall pub.ai \sqcap \forall pub.acm\_trans$ as input query in Example 5.1. Applying the algorithm of Figure 3 we obtain:

$$
\begin{array}{l}
\mathcal{D}(Q', \mathcal{P}) \\
\quad = \mathcal{S}_{inf}(\forall pub.ai, \mathcal{P}) \sqcap \mathcal{S}_{inf}(\forall pub.acm\_trans, \mathcal{P}) \\
\quad = (\ Sol(K) \sqcup (Sol(H) \sqcap Sol(J))\ ) \ \sqcap \ Sol(I)
\end{array}
$$

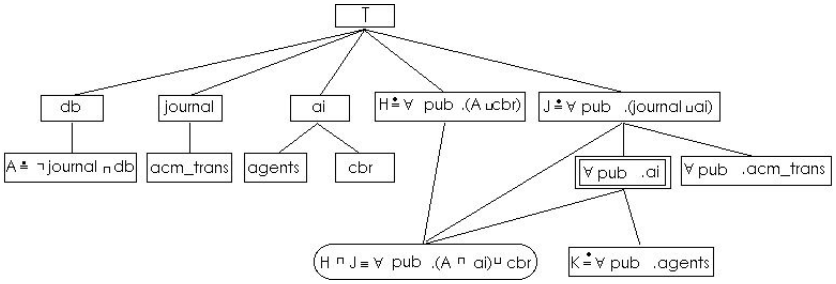

Figure 2: An example of terminology.

```
Function  D(Q : concept; P : terminology) : concept;
begin
    if      Q is a primitive condition or Q ∈ P then return S_inf(Q, P)
    elseif  Q ≡ C ⊔ D      then if  D(C, P) ≠ ⊥ and D(D, P) ≠ ⊥
                                then return D(C, P) ⊔ D(D, P)
                                else return S_inf(Q, P)
    elseif  Q ≡ C ⊓ D      then if  D(C, P) ≠ ⊥ and D(D, P) ≠ ⊥
                                then return D(C, P) ⊓ D(D, P)
                                else return S_inf(Q, P)
    elseif  Q ≡ ¬C         then if  D(C, P) ≠ ⊥
                                then return ¬Dual_D(C, P)
                                else return S_inf(Q, P)
    elseif  Q ≡ ∀R.C       then if  D(C, P) ≠ ⊥
                                then return ∀R.D(C, P)
                                else return S_inf(Q, P)
    elseif  Q ≡ ∃R.C       then if  D(C, P) ≠ ⊥
                                then return ∃R.D(C, P)
                                else return S_inf(Q, P)
    elseif  Q ≡≤ nR        then return (∀R.S_inf(⊤, P)) ⊓ (≤ nR)
    elseif  Q ≡≥ nR        then return (∀R.S_inf(⊤, P)) ⊓ (≥ nR)
    elseif  Q ≡ (R1 = R2)  then return (∀R1.S_inf(⊤, P)) ⊓ (∀R2.S_inf(⊤, P)) ⊓ (R1 = R2)
    else return ⊥
end;
```

Figure 3: The decomposition algorithm.

Notice that Condition (1) is preserved by above reuse methods since it is easy to prove (by induction) that the following theorem holds.

Theorem 6.1 *Let Q be a query. Let $\mathcal{P}$ be the terminology of a given case memory. Then*

$$\mathcal{S}_{inf}(Q,\mathcal{P}) \sqsubseteq Q \qquad\qquad \mathcal{D}(Q,\mathcal{P}) \sqsubseteq Q$$

Notice that in general $\mathcal{S}_{inf}(Q,\mathcal{P}) \neq \mathcal{D}(Q,\mathcal{P})$. As a consequence, we can combine the above methods in order to obtain a wider solution, i.e., $Sol(Q) \doteq \mathcal{S}_{inf}(Q,\mathcal{P}) \sqcup \mathcal{D}(Q,\mathcal{P})$

7 Distributed Query Retrieval and Reuse

A *local failure in query reuse* occurs when a mediator is not able to rewrite a given query Q, i.e., $Sol(Q) = \bot$[1]. This means that the mediator's case memory contains no past cases that can be used to reformulate Q. Usually this is due to the fact that it is the first time that the consumer formulates such a query, i.e., the consumer has a new information need.

Example 7.1 Let us consider the case memory of Example 5.1 and the query $Q'' \doteq \forall pub.ai \sqcap \exists affiliation.\{\text{“}Stanford\text{”}\}$. The mediator is not able to rewrite the query. Indeed, we have

$$\mathcal{D}(Q'',\mathcal{P}) = (\ Sol(K) \sqcup (Sol(H) \sqcap Sol(J))\ ) \sqcap \bot = \bot$$

Furthermore a *local failure in query evaluation* occurs when a mediator send a rewritten query to related sources and receives at least an empty answer. This means that the case memory of the mediator is not updated. Typically, an information source has been removed from the system or changed its schema.

Example 7.2 Let us suppose that source w_1 has been (perhaps temporarily) removed, then the evaluation of the reformulated query in Example 6.3 fails. Indeed $I_{w_1}(\forall pub.conference) = I_{w_1}(\forall pub.acm\_trans) = I_{w_1}(\forall pub.journal) = I_{w_1}(\forall pub.\exists keyword.\{\text{“}Agents\text{”}\}) = \emptyset$

Even if a local failure (in query reuse or evaluation) occurs, the system has still a possibility of solving the

[1] Notice that $I(\bot) = \emptyset$.

problem by means of distributed query retrieval and reuse. Notice that this provides the system the most important way of learning. Indeed, if the query is reformulated, the new rewritten query and/or the new sources can be stored as a new case. In such a way the mediator can support dynamic information systems. Furthermore, it does not need to maintain consistent its case memory every time something changes, but only when a consumer sends a query that fails. This allows us to avoid of overloading the distributed information system with consistency maintenance operations, that usually are time consuming. Distributed query retrieval and reuse is based on cooperation with other mediators and sources. Cooperation strategies can be classified according to three parameters: partners, queries, and answers.

- **Partners.** When a mediator (say M) fails, it can cooperate with other *mediators* asking them to rewrite a query according to their own case memories. If they succeed, M can store the result as a new case in its case memory. The mediators involved in this strategy can be all the mediators of the system or just mediators that have never been in touch with M. The mediator M can directly cooperate with information sources as well. It can involve all the sources (it is very expensive), sources responsible of the local failure (this strategy takes into account changes of schemas), or recently added sources (this strategy takes into account new sources). Each source processes the query with an algorithm based on maximally contained rewriting (it is the Algorithm 2, but the schema is static instead of dynamic) and returns the answer.

- **Queries.** Cooperation strategies can also be classified according to the query. Indeed M can send the original query or the reformulated query (if any). In the latter case the "approximation of the solution grows". For example, let us suppose that Q is the original query and $Q' = Sol_M(Q)$ is the query rewritten by M. Let N be the mediator or source that cooperates with M and thus receives Q' as input query. Let us suppose that $Q'' = Sol_N(Q')$ is the query rewritten by N. The Theorem 6.1 states that $Q'' \sqsubseteq Q' \sqsubseteq Q$. As a consequence, Q'' is more distant from Q than Q'

  Cooperation strategies can also be classified depending on the fact that the query can be sent as it is or be decomposed in basic components.

- **Answers.** Finally, cooperation strategies can be classified according to the answer. M can ask for rewriting the query. Its goal is to update its case memory with a rewritten query obtained from its collaborators. M can also ask for data that answer the query. In such a situation, it stores in the case memory the addresses of the mediators/sources that answered.

Let us focus on two of the possible strategies.

As a first strategy, let us consider a mediator M that cooperates with the others mediators, sends them the original query, and asks for receiving the rewritten query. A possible disadvantage of such a strategy is that asymptotically all the mediators may have the same case memory. This result is a consequence of the following theorem.

Theorem 7.1 *Let M, N be two mediators such that M interacts with N when M fails. Let $\mathcal{C}_n(M)$ be the case memory of M after n interactions with N. Let $\mathcal{C}(N)$ be the case memory of N such that it does not change while N interacts with M. Then*

$$\lim_{n\to\infty} \frac{card(\mathcal{C}_n(M) \cap \mathcal{C}(N))}{card(\mathcal{C}(N))} = 1 \qquad (3)$$

The above theorem is quite straightforward to prove and states that the case memory of M converges to the case memory of N when M cooperates with N.

As a second strategy, let us consider a mediator M that directly cooperates with information sources, sends them the original query, and receives both the rewritten query and the data. This strategy guarantees a given consumer that the mediator M converges to the consumer's information need (i.e., the recall ratio converges to 1). Indeed it is possible to prove the following theorem.

Theorem 7.2 *Let $S_1, \ldots, S_n$ be n information sources. Let $\mathcal{V}$ be a view of $S_1, \ldots, S_n$. $\mathcal{V}$ is represented as a case memory that does not change. Let M be a mediator such that M interacts with $S_1, \ldots, S_n$ when it fails. Let $\mathcal{C}_n(M)$ be the case memory of M after n interactions with $S_1, \ldots, S_n$. Then*

$$\lim_{n\to\infty} \frac{card(\mathcal{C}_n(M) \cap \mathcal{V})}{card(\mathcal{V})} = 1 \qquad (4)$$

The theorem above states that if the information need of a consumer is represented by $\mathcal{V}$, then the mediator will asymptotically satisfy such a need. Let us demonstrate the utility of this strategy with an example.

Example 7.3 In the situation of Example 7.2, let us suppose that the mediator chooses to look for new sources and cooperate with them. Let us suppose that w_4 is a new information source which contains ACM publications and has the following schema:

$$\ldots$$
$$acm\_tods \doteq \forall j\_name.\{\text{"}TODS\text{"}\} \sqcap \forall publisher.\{\text{"}ACM\text{"}\}$$
$$acm\_tocl \doteq \forall j\_name.\{\text{"}TOCL\text{"}\} \sqcap \forall publisher.\{\text{"}ACM\text{"}\}$$
$$acm\_tois \doteq \forall j\_name.\{\text{"}TOIS\text{"}\} \sqcap \forall publisher.\{\text{"}ACM\text{"}\}$$
$$\ldots$$

The mediator decomposes the query Q' and sends each component to w_4. For example, when w_4 receives $\forall pub.acm\_trans$, it looks for maximally contained concepts and obtains $M_{inf}(\forall pub.acm\_trans, \mathcal{P}) = \{\forall pub.acm\_tods, \forall pub.acm\_tocl, \forall pub.acm\_tois\}$. When the mediator receives the answer, retains as new case:

$$\langle\ \forall pub.acm\_trans\ ,\ \forall pub.acm\_tods \sqcup \forall pub.acm\_tocl \sqcup \forall pub.acm\_tois\ ,\ \langle\ (\forall pub.acm\_tods, w_4)\ ,\ (\forall pub.acm\_tocl, w_4)\ ,\ (\forall pub.acm\_tois, w_4)\ \rangle\ \rangle$$

8 Conclusions

We considered the problem of rewriting queries by means of distributed case-based reasoning. As far as we know, this is a novel approach. We showed that the notion of maximally contained rewriting of a query can be also applied to a case memory (what we call local query retrieval). This allows us to have dynamic mediated schemas instead of static ones. We also showed that a real dynamic mediator is obtained by means of distributed query retrieval and reuse, i.e., cooperation with other mediators and sources. Thanks to this approach, the mediator is able to be updated when sources and consumers are added/removed or change their schemas/needs. This operation is performed only when a consumer sends a query that fails and not every time something changes. This allows us to avoid of overloading the distributed information system with consistency maintenance operations, that usually are time consuming. In distributed case-based reasoning, the choice of the right cooperation strategy is crucial. We hinted several possible strategies and sketched a discussion about their advantages. We also demonstrated by examples the utility of this approach. Finally, notice that the choice of an appropriate description logic can reduce the algorithm complexity for query retrieval and reuse. For example, if we choose *core*–CLASSIC [3], we have that the maximally-contained rewriting can be computed in polynomial time.

Acknowledgement

This work was partially supported by M.U.R.S.T. under Project INTERDATA. We thank J. Mylopoulos for suggestions and useful discussions about the topics presented in this paper. Comments by P. Coupey, C. Diamantini, and L. Penserini are also greatly appreciated.

A Description Logic

A description logic is composed by symbols taken from the alphabet of Concept Names, the alphabet of Role Names, and the alphabet of Individual Names (or Individuals). It also includes a set of constructors that permit the formation of Concepts and Roles (see Table 1). Its formal semantics (e.g., [6]) is the structure $\mathcal{I} = (\Delta, I())$. It consists of a nonempty set Δ (the domain of $\mathcal{I}$) and a function $I()$ (the interpretation function of $\mathcal{I}$) that maps every individual to an element of Δ, every concept to a subset of Δ, and every role to a subset of $\Delta \times \Delta$ (see Table 1). We say that C is subsumed by D iff $I(C) \subseteq I(D)$ for every interpretation $\mathcal{I}$. A *terminology* has statements about concepts (see Table 1) and intuitively describes the conceptual schema of a database.

References

[1] Ahmed and et al. The Pegasus heterogeneous multidatabase system. *IEEE Computer*, 24(12), 1991.

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

[3] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting Queries Using Views in Description Logics. In *Proc. of the 16th ACM Symposium on Principles of Database Systems (PODS)*, pages 99–108, New York, NY, May 12–14, Tucson, Arizona 1997. ACM Press.

[4] A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[5] R.J. Brachman and J.G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9:217–260, 1985.

[6] M. Buchheit, F. M. Donini, and A Schaerf. Decidable Reasoning in Terminological Knowledge Representation Systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th International Conference on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[8] R.G.G. Cattell. *The Object Database Standard ODMG-93*. Morgan Kaufmann, San Mateo, CA, 1993.

[9] S. R. Chaudhuri, S. R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *International Conference on Data Engineering*, 1995.

Constructor Name	Syntax	Semantics
concept name	A	$I(A) \subseteq \Delta$
top	$\top$	Δ
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$I(C) \cap I(D)$
disjunction	$C \sqcup D$	$I(C) \cup I(D)$
negation	$\neg C$	$\Delta \setminus I(C)$
universal quantification	$\forall R.C$	$\{s \mid \forall t.(s,t) \in I(R) \rightarrow t \in I(C)\}$
existential quantification	$\exists R.C$	$\{s \mid \exists t.(s,t) \in I(R) \wedge t \in I(C)\}$
number restrictions	$\leq nR$	$\{s \mid card(\{t \mid (s,t) \in I(R)\}) \leq n\}$
	$\geq nR$	$\{s \mid card(\{t \mid (s,t) \in I(R)\}) \geq n\}$
equality restriction	$Q = R$	$\{s \mid \{r \mid (s,r) \in I(Q)\} = \{t \mid (s,t) \in I(R)\}\}$
role name	P	$I(P) \subseteq \Delta \times \Delta$
role chain	$Q \circ R$	$\{(s,t) \mid \exists r.(s,r) \in I(Q) \wedge (r,t) \in I(R)\}$

Statement Name	Syntax	Semantics
Concept Specification	$A \dot{\leq} C$	$I(A) \subseteq I(C)$
Concept Definition	$A \doteq C$	$I(A) = I(C)$

where A is a concept name, C, D are concepts, P is a role name, and Q, R are roles.

Table 1: Syntax and semantics of description logics.

[10] W. W. Chu and G. Zhang. Associative query answering via query feature similarity. In *International Conference on Intelligent Information Systems (IIS'97)*, The Bahamas, 1997.

[11] C. Collet, M. N. Huhns, and W.-M. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, 1991.

[12] P. Coupey, C. Fouqueré, and S. Salotti. Formalizing partial matching and similarity in CBR with a description logic. *Applied Artificial Intelligence*, 12(1):71–112, 1998.

[13] J. J. Daniels and E. L. Rissland. A Case-Based Approach to Intelligent Information Retrieval. In *Proc. of the SIGIR'95 Conference*, pages 238–245, Seattle, WA, 1995. ACM.

[14] C. Diamantini and M. Panti. A Conceptual Indexing Method for Content-Based Retrieval. In A. M. Tjoa, A. Cammelli, and R. R. Wagner, editors, *Tenth International Workshop on Database and Expert Systems Applications (DEXA'99)*, Florence, Italy, 1–3 September 1999. IEEE Computer Society.

[15] C. Diamantini, M. Panti, L. Spalazzi, and S. Valenti. Federated Information System Architectures for Local Public Administration. In M. Khosrowpour, editor, *Information Resource Management Association International Conference (IRMA'99)*, Hershey, PA, May 1999. Idea Group.

[16] O. M. Duschka and M. R. Genesereth. Infomaster - An Information Integration Tool. In *Proc. of the International Workshop on Intelligent Information Integration*, Freiburg, Germany, September 1997.

[17] O. M. Duschka and M. R. Genesereth. Query Planning with Disjunctive Sources. In *Proc. of the AAAI Workshop on Artificial Intelligence and Information Integration*, Madison, WI, July 1998.

[18] G. Fouqué, W. W. Chu, and H. Yau. A case-based reasoning approach for associative query answering. In *Proc. of the 8th International Symposium on Methodologies for Intelligent Systems*, pages 183–192, Charlotte, N.C., October 1994.

[19] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems*, 8:117–132, 1997.

[20] A. Giretti and L. Spalazzi. ASA: A Conceptual Design-Support System. *Engineering Application of Artificial Intelligence*, 10(1):99–111, January 1997.

[21] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The Stabford data warehousing project. *IEEE Bull. on Data Engineering*, 18(2):3–18, 1995.

[22] G. Kamp. Using Description Logics for Knowledge Intensive Case-Based Reasoning. In *Proc. of th 3rd European Workshop on Case-Based Reasoning*, volume 1168 of *Lecture Notes in Artificial Intelligence*, Berlin, Germany, 1996. Springer-Verlag.

[23] W. Kim and et al. On Resolving Semantic Heterogeneity in Multidatabase Systems. *Distributed and Parallel Databases*, 1(3), 1993.

[24] T. A. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *AAAI Spring Symposium on Information Gathering*. AAAI, 1995.

[25] Jana Koehler. An Application of Terminological Logics to Case-based Reasoning. In Pietro Torasso, Jon Doyle, and Erik Sandewall, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 351–362, Bonn, Germany, May 1994. Morgan Kaufmann.

[26] D. Leake, editor. *Case-Based Reasoning : Experiences, Lessons, & Future Directions*. AAAI Press / The MIT Press, 1996.

[27] F. Manola and et al. Ditributed object management. *International Journal of Intelligent and Cooperative Information Systems*, 1(1), March 1992.

[28] T. Ozsu, U. Dayal, and P. Valduriez. *Distributed Object Management*. Morgan Kaufmann, San Mateo, CA, 1993.

[29] M. V. N. Prasad, V. R. Lesser, and S. E. Lander. Retrieval and reasoning in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 7(1):74–87, March 1996.

[30] S. Ram. Special issue on heterogenous distributed database systems. *IEEE Computer Magazine*, 24(12), December 1991.

[31] C. Ramirez. Case-based reasoning applied to information retrieval. In *IEEE Coloquium on Case-Based Reasoning: Prospects for Application*, London, February 1995. IEE.

[32] A. Sheth. Special Issue in Multidatabasem Systems. *ACM SIGMOD Record on Management of Data*, 20(4), December 1991.

[33] A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Transaction on Database Systems*, 22(3), 1990.

[34] J. D. Ullman. Information Integration Using Logical Views. In *Proceedings of the International Conference on Database Theory*, pages 19–40, 1997.

[35] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer Magazine*, 25:38–49, March 1992.

[36] H. Z. Yang and A. Larson. Query transformation for PSJ queries. In *Proc. of the International Conference on Very Large Data Bases*, pages 245–254, 1987.

[37] G. Zhou, R. Hull, and R. King. Generating data integration mediators that use materializations. *J. of Intelligent Information Systems*, 6:199–221, 1996.