A Framework for Semantic Grouping in P2P Databases*

Verena Kantere, Dimitrios Tsoumakos and Timos Sellis

School of Electr. and Comp. Engineering, National Technical University of Athens,
vkante@dbnet.ece.ntua.gr,
dtsouma@cslab.ece.ntua.gr,
timos@dbnet.ece.ntua.gr

Abstract. Sharing of structured data in decentralized environments is a challenging problem, especially in the absence of a global schema. Social network structures map network links to semantic relations between participants in order to assist in efficient resource discovery and information exchange. In this work, we propose a scheme that automates the process of creating schema synopses from semantic clusters of peers which own autonomous relational databases. The resulting mediated schemas can be used as global interfaces for relevant queries. Active nodes are able to initiate the group schema creation process, which produces a mediated schema representative of nodes with similar semantics. Group schemas are then propagated in the overlay and used as a single interface for relevant queries. This increases both the quality and the quantity of the retrieved answers and allows for fast discovery of interest groups by joining peers. As our experimental evaluations show, this method increases both the quality and the quantity of the retrieved answers and allows for faster discovery of semantic groups by joining peers.

1 Introduction

In the last few years, there has been a growing interest in the Peer-to-Peer (P2P) computing paradigm, primarily boosted by popular applications that enable massive data sharing among millions of users. The P2P paradigm dictates a fully distributed, cooperative network design, where nodes collectively form a system without any supervision. Many popular P2P applications (e.g., Gnutella [15]) operate on unstructured networks, with peers joining and leaving the system in an ad-hoc fashion, while maintaining only local knowledge. While structured overlays (e.g., [39]) provide efficient lookup operations, in many realistic scenarios the topology cannot be controlled and thus they cannot be used (e.g., dynamic ad-hoc networks or existing large-scale unstructured overlays).

In the variety of P2P applications that have been proposed, Peer Data Management Systems (PDMSs) (e.g., [17, 41]) hold a leading role in sharing semantically rich information. In a PDMS, each peer is an autonomous source that has a local schema. Sources store and manage their data locally, revealing part of their schemas to the rest of the peers. Due to the lack of global schema, they express and answer queries based on their local schema. Peers also perform local coordination with their *acquaintees*, i.e., their one-hop neighbors in the overlay. During the acquaintance procedure, the two

* Extended version of a paper to appear in DEXA 2007.

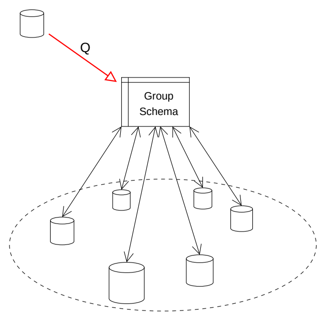

Fig. 1. Query directed towards a group schema

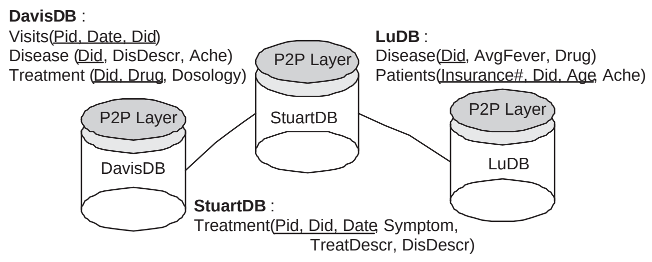
DavisDB :
Visits(Pid, Date, Did)
Disease (Did, DisDescr, Ache)
Treatment (Did, Drug, Dosology)
LuDB :
Disease(Did, AvgFever, Drug)
Patients(Insurance#, Did, Age, Ache)
P2P Layer
StuartDB
P2P Layer
DavisDB
P2P Layer
LuDB
StuartDB :
Treatment(Pid, Did, Date, Symptom, TreatDescr, DisDescr)

Fig. 2. Part of a P2P system with peer-databases from the health environment

peers exchange information about their local schemas and create mediating mappings semi-automatically [23]. The establishment of an acquaintance implies an agreement for the performance of data coordination between the acquaintees based on the respective schema mapping. However, peers do not have to conform to any kind of data or schema transformation to establish acquaintances with other peers and participate in the system. The common procedure for query processing in such a system is the propagation of the query on paths of bounded depth in the overlay. At each routing step, the query is rewritten to the schema of its new host based on the respective acquaintance mappings. A query may have to be rewritten several times from peer to peer till it reaches nodes that are able to answer it sufficiently in terms of quality but also quantity.

Several theoretical frameworks have been proposed for PDMSs [9,14,17,36]. These frameworks aim at the provision of correct and complete semantics that distinguish PDMSs from data integration systems and also handle inconsistency in such a way that the system does not collapse in its presence. The works in [9, 14] employ epistemic and autoepistemic logic in order to achieve maximum peer autonomy w.r.t. both the local semantics and the peer connectivity. Even though we acknowledge the efficiency of these results, in this work we adopt a PDMS framework towards the lines of [17, 36], since *First Order Logic* and the *relational model* are more convenient for a practical PDMS.

Assuming a social network organization in a PDMS, an interesting question is how to automatically create a synopsis of the common interests of a group of semantically related nodes. This will be a mediating schema representative of the group along with its mappings with the local databases. Queries can then be expressed on this mediated schema (see Figure 1). This functionality is desirable for multiple reasons:

First, it allows queries to be directed to a single, authoritative schema. In this way query answers are more precise, since they come from the most relevant peers. Moreover, these peers receive a query version that has not been rewritten successively multiple times, but only twice (i.e. from the original query to the group schema and from the group schema to the peer that will answer the query), and most importantly, through a mediating schema that is as lossless as possible in terms of semantics. Thus, the query version that the relevant peers receive is not degraded so much as it is through multiple successive rewritings [25, 41]. Furthermore, using the group schema as the mediator of query answering, is much less time-consuming.

Second, the group schema actively expedites the acquaintance between semantically related peers. It allows joining peers with little or no memory to be promptly and favorably interconnected. Since group schemas can be periodically advertised inside the network, each newcomer is able to make an "educated guess" as to which groups are interesting to him. Hence, users with no prior exposure to the different (and possibly numerous) schemas of remote nodes will save the time and bandwidth that a learning process requires.

Finally, it minimizes human involvement in the process of creating/updating the group schema. Until now, nodes have been organized by means of a human-guided process (usually by one or more administrators and application experts) into groups of peers that store semantically related data. The administrator, using schema matching tools as well as domain knowledge, initiates and maintains these synopses. This approach requires manual work, extensive peer coordination and repetition of this process each time the group changes.

The above advantages of group schema mediation in a PDMS would be unimportant if group schemas do not conform without major overhead to the vital P2P features of autonomy and dynamicity. Thus, group schemas should be created and maintained dynamically and in an automatic way. Moreover, group schemas should be adaptable to the changes of semantics of the peers that participate in the PDMS. Therefore, they should be able to evolve accordingly to the peer joining, leaving, or interest changing.

Motivating example

As a motivating example, envision a P2P system where the participating peers are databases of private doctors of various specialties, diagnostic laboratories and databases of hospitals. Figure 2 depicts a small part of this system, where the peer databases (or else, pDBMSs) are: DavisDB - the database of the private doctor Dr. Davis, LuDB - the database of pediatrist Dr Lu and StuartDB - the database of the pharmacist, Mr Stuart. A P2P layer, responsible for all data exchange of a peer with its acquaintees, sits on top of each database. Among others, the P2P layer is responsible for the creation and maintenance of mappings of local schemas during the establishment of acquaintances towards the line of [23]. Moreover, each peer owns a query rewriting and a query-schema matching mechanism. The schemas of the databases are shown in Figure 2.

We would like to automatically produce a merged schema for all three peers of our example, semantically relevant to their local schemas. Such a merged schema could be the following:

Disease/Sickness($\underline{\text{Did}}$, DisDescr, Symptom, Drug)

Visits/Patients($\underline{\text{Pid/Insurance\#}}$, $\underline{\text{Did}}$, $\underline{\text{Date}}$, Age, Ache)
Treatment($\underline{\text{Did}}$,$\underline{\text{Drug}}$, Dosology)

Obviously, in the merged schema we would like alternative names for relations or attributes (separated by '/' above). We would also like the merged schema to contain relations or attributes according to their frequency in the set of local schemas. For example, the attribute *Patients.AvgFever* is not present, possibly because the respective concept is not considered to be frequent in the set of local schemas.

In this paper, we describe a mechanism that operates on a semantically clustered flat (i.e. without super-peers) PDMS and automatically creates relational schemas that are representative of the existing clusters. Given the semantic neighborhoods, our system can initiate the creation of a mediating schema $\mathcal{S}_{\mathcal{G}}$ that summarizes the semantics of the participating database schemas. It is created by the gradual merging of peer schemas along the path followed by the process. We call *interest* or *semantic groups* the semantic clusters that exist in social networks operating on PDMSs; moreover, we call *group schema* the inferred schema of the group $\mathcal{S}_{\mathcal{G}}$. $\mathcal{S}_{\mathcal{G}}$ holds mappings with each of the peers involved in its creation and functions as a point of contact for all incoming queries, whether from inside or outside the semantic neighborhood. Thus, requesters of information need only maintain mappings and evaluate queries against one schema, instead of multiple ones. The inferred groups are advertised after their creation and are not managed by any specific peer. Furthermore, the inferred schemas are periodically broadcast in the overlay, so that joining peers can direct their queries or participate in groups similar to their interests. Group schemas are created and managed automatically so that they are dynamically adapted to the change of peer semantics, due to peer join and leave, as well as change in peer needs for information. Thus, group schemas are always representative of peer semantics, without any overhead of human interaction. Our experimental evaluation shows that our group creation process increases both the accuracy and the number of answers compared to individually propagating and answering queries in an unstructured PDMS.

In Section 2 we describe the basic notions of the framework that we consider and give some essential formal definitions. In Section 3 we describe the core characteristics of the inference procedure of the group schema. In Section 4 we present our experimental results and Section 5 summarizes related work. Finally, Section 6 concludes the paper.

2 Preliminaries

We assume a PDMS with a social-network organization of peers, i.e., semantically relevant peer DBMSs are acquainted or close in the overlay. This can be achieved either manually or using one of the proposed schemes (e.g., [25, 34]). Peer schemas are relational, (i.e., the only internal mappings are foreign key constraints). Acquainted peers create and maintain schema mappings between them that are of the widely-known GAV/LAV/GLAV form [17, 27]. A mapping M that refers to the schemas S_1, S_2 of the acquainted peers peers p_1, p_2, respectively, is stored locally in both of them. Peers p_1 and p_2 can employ M in order to rewrite a query that is expressed on their local schema (S_1, S_2, respectively) on the schema of the other peer, (S_2, S_1, respectively). Moreover, peers do not carry additional semantic information about their schemas and mappings.

In our setting, semantics of peer schemas and data, are derived solely from the local schemas, data and mappings between acquainted peers. We define a distinct concept of a schema S to be each element $R.A$, where A is an attribute of relation R of schema S:

Definition 1. *Considering a relational schema S, a distinct concept corresponds to each $R.A$ where A is an attribute of relation $R \in S$.*

A schema mapping between peers is actually a a set of concept correspondences that hold under a set of conditions. The conditions are either attribute joins or attribute value constraints. The following is the definition of a mapping:

Definition 2. *Considering a source schema S and a target schema S', a GAV/LAV/GLAV mapping between them $M(S,S')$ is the set $\{Cr_M(S,S'),Cond_M(S,S')\}$, where the set of concept correspondences $Cr_M(S,S') = \{R.A = R'.A' | R.A \in S, R'.A' \in S'\}$ holds under the set of conditions $Cond_M(S,S') = \{R_1.A = R_2.B$ or $R_1.A = const | R_1,R_2 \in S$ or $R_1,R_2 \in S'\}$; const is a data value.*

Obviously, for each pair of concepts $\{R.A, R'.A'\}$ that each belong to a different schema, $R.A \in S$ and $R'.A' \in S'$, and that are corresponded through a mapping $M(S,S')$, there is one such pair in $Cr_M(S,S')$. A set of mappings between S,S' is denoted as $\mathcal{M}(S,S')$.

Semantics are 'flooding' from one peer to the other, through respective mappings:

Definition 3. *For each correspondence $R.A = R'.A' \in Cr_M(S,S') \in M(S,S')$, the concepts, $R.A$, $R'.A'$ are considered equivalent.*

Extended discussion and details on acquaintance mappings, query rewriting, query similarity etc can be found in [25].

3 Interest Group Creation

Our goal in creating a group schema is to represent the semantic clusters in a social network using a distributed process that iteratively merges local schemas into the final group schema that preserves their most frequent semantics.

In social network systems, nodes with relevant information are close in overlay distance. Yet, this semantic clustering is implicit, in that peers have no knowledge of the number of the participants and their common characteristics. The need for explicit knowledge of the semantic groups spread over a network has multiple advantages: First, it enables peers to direct relevant queries promptly towards "authority" nodes. In many distributed systems, new peers join the network using random entry points. Therefore, they would like to be informed about the various semantic groups in which they could participate and select acquaintees from. In addition, since most such systems exhibit a highly dynamic behavior, with node arrivals/departures and possible schema or workload changes, meta-data on semantic groups can be refreshed. Nevertheless, clustering w.r.t. the semantics of all peers requires constant maintenance whereas group maintenance (w.r.t. only the group semantics) can be performed occasionally. Thus, it is easier to maintain semantic groups than implicit semantic clusters. Nevertheless, it is essential to the social network that these semantic groups are dynamic, in order to follow the evolution of the content and structure of the overlay.

In the following we describe the inference procedure of explicit semantic groups in social networks. Furthermore, we encounter the implications of multiple concurrent or sequential inferences of distinct groups and we discuss management methods.

3.1 Group Inference

In this section we describe the process through which a group schema emerges from a set of clustered nodes in our system. The group-inference procedure comprises of the following steps:

– Initialization: Who and when initiates the group schema inference
– Propagation: How does the process advance among peers of the same group
– Termination and Refinement: When is the process over/reiterated

Figure 3 summarizes the above steps that are described extensively in the following.

Group_Inference_Procedure
Input: The prospective initiator peer I
Output: The group schema S_{IG}
Initialization: $P = I$, i.e. $ST(I) = \emptyset$, $S_{IG} = \emptyset$, #visitedpeers = 0, $S_P = S_I$, $S_{IG} = \emptyset$, $\mathcal{M} = \emptyset$, $\mathcal{M}_i = \emptyset$, $D = \emptyset$.
Step1: If I is not *active* abort the procedure.
Step2: Add appropriate acquaintees of P to $ST(I)$ and reorder the latter.
Step1: Augment #*visitedpeers* by one.
Step3: Call $\{S'_{IG}, \mathcal{M}'_i, D'\}$ = ***Schema_Merging_Algorithm(***S_{IG}**,** S_P**,** $\mathcal{M}$**,** $\mathcal{M}_i$**,** D***)***.
Step4: If (#visitedpeers =$MaxP$) then if the $DoC(S'_{IG}, S_{IG})$ is below a certain threshold, then goto Step7; else goto Step5.
Step5: Select the next peer, P', that corresponds to the next element in the stack $ST(I)$. Set $\mathcal{M}$ equal to the set of mappings between P, P'; change attribute and relation names of P in $\mathcal{M}'$ to the respective names in the merged schema S'_{IG}.
Step6: Set $S_{IG} = S'_{IG}$, $P = P'$ go to Step2.
Step7: Return S_{IG}.

Fig. 3. Group inference procedure

Initialization

The nature of our application requires that the group inference is performed in a distributed manner, without global coordination. Peers should be able to start the process that creates the respective schema with minimum message exchange. In our system, each member of the social group is eligible to initiate the inference process. Nevertheless, such groups may consist of numerous participants resulting in very frequent collisions among competing initiators. Hence, we only allow *active* members to become the initiators of the process. This is enforced by a system-wide parameter that defines the minimum number of queries posed in the most recent time frame. Intuitively, active peers have a better knowledge of the social network and the schemas of the other participants through the answers they receive. Moreover, since the network is semantically clustered, the active peers have good knowledge of the semantic cluster(s) they belong to.

The initiator's local schema becomes a point-of-reference regarding the inferred one. Thus, the peer schemas considered for the formation of the group schema should not differ semantically a lot from the schema of the initiator. Specifically, we require that the participating local schemas should be at least t-similar to the initiator's schema: t is a parameter that mainly determines how specialized (only peers very similar to the initiator considered) or general (a broad collection of peers participate in the process) the inferred schema will be. The initiator peer is called the *originator* of the group, its schema is the *origin* of the group schema and the maximum similarity distance between the origin and the peer schemas that participate in the group schema inference is the *semantic radius* of the group.

The following function calculates the *directed* semantic similarity, SS, of two relational schemas:

$$SS(S,T) = \frac{\sum_i \sum_j w_{ij} Mapped_T(SR_{ij})}{\sum_i \sum_j w_{ij} SR_{ij}} \tag{1}$$

In function 1, SR_{ij} is the j^{th} attribute of the i^{th} relation of S. S is the source schema and T is the target schema. SS calculates the portion of S's attributes (SR) that are mapped on T. Specifically, SS is the average sum of the weighted sum of all the attributes of S that correspond to attributes of T through mappings. Thus, $Mapped_T(SR_{ij})$ is a boolean function that gives 1 if SR_{ij} is mapped to some attribute in schema T, or it gives 0, otherwise. Also, $w_{ij} > 1$ for attributes of S that belong to relation keys and $w_{ij} = 1$ otherwise. Obviously, $SS(S,T) \neq SS(T,S)$ in general. SS achieves to measure semantic similarity because it takes into consideration the mapping of concepts beyond their structural interpretations on the schema level. Moreover, since SS ignores the schema structure, it is very easily calculated.

Propagation

The initiator I, with schema S_I of the inference process initializes the group schema to its own and creates a stack $ST(I)$ with its acquaintees that are part of the cluster. Specifically, $ST(I) = \{A_1, A_2, ..., A_m\}$ is an ordered set of elements $A_j = \{P_j, SS(S_I, S_{P_j})\}$, where P_j is a peer with schema S_{P_j}. Elements A_j refer to the I's most similar acquaintees: $SS(S_I, S_{P_j}) \geq t,\ j = 1,..,m$ and $SS(S_I, S_{P_j}) \geq SS(S_I, S_{P_{j+1}}),\ j = 1,..,m-1$. The initiator propagates the inference procedure to the first peer on the stack. The latter is supposed to merge its own schema with the group schema it receives according to the merging procedure described in the next subsection. The stack ST dictates a network path that corresponds to the peer order of the stack ST(I). Every peer P (beyond the initiator) on the network path that the inference process follows, determines its acquaintees P_j for which $SS(S_I, S_{P_j}) \geq t$, adds the respective pair P_j, $SS(S_I, S_{P_j})$, to $ST(I)$ and orders the latter. Any peer P on the inference process path calculates $SS(S_I, S_{P_j})$ indirectly, as the product: $SS(S_I, S_P) \cdot SS(S'_P, S_{P_j})$, where S'_P is the part of S_P mapped on S_I. Essentially, $SS(S_I, S_P)$ aims to measure how much of the semantics of S_I can be found on schema S_P, independently of other semantics that the latter captures. The only way to measure this (without automatic matching) is through the chain of mappings of S_I all the way to S_P. As such, the value of $SS(S_I, S_P)$ depends on the path that the inference process follows and fails to consider concepts that exist both in S_I and S_P but did not exist in the schemas of intermediate nodes. However, this formula produces a satisfactory result, since nodes are visited in decreasing order of similarity with I and clustering precedes this process, so a peer P will have higher similarity with the originator than successor nodes in the stack [1]. Moreover, if a peer P already in $ST(I)$ is considered for addition, the entry with the highest $SS(S_I, S_P)$ value is kept.

Even though the participation or not of peers in the inference process is judged by a part of their schemas, their whole schema contributes to the inferred group schema (see subsection 3.2). Intuitively, the goal of the inference process is to produce a schema that represents semantics encapsulated in the cluster. In order to determine the cluster's semantic borders we use as a reference the semantics of the initiator. In this way the process is safe from producing a schema too much broader or distorted from the interests of the initiator.

Termination

As aforementioned, the group inference procedure ends when the stack of participating peers becomes empty. However, if too many peers own schemas very similar to the originator's schema or the similarity threshold t is too small (i.e. the semantic radius of the inferred group is big), then it may be the case that the stack is provided at each step with a lot of new entries. Thus, the inference procedure is prolonged taking into account a big number of peers. After a certain number of iterations, there is usually no point of considering more peer schemas in the inference procedure, because they do not alter the schema significantly. In order to reduce the time of the inference and save the network from spending resources on pointless iterations of the procedure, we add a limit to the maximum number of encountered peer schemas, *MaxP*, as a termination condition. A small *MaxP* value implies the desire for more specialized groups. *MaxP* is not a *Time To Live* (hereafter TTL) condition, since successive hops are not always on the same path; *MaxP* refers to the total number of participating nodes and not just the nodes on one path.

Finally, there may be situations where the inference procedure terminates due to *MaxP* while important semantic information is still added, or continues until *MaxP* is reached while little information is assimilated. To rectify this, we also consider the *degree of change*, or else *DoC*, that occurs to the inferred schema during each merging step. Note that this threshold should be taken into account for termination after *MaxP* is reached. Otherwise, in case of a well clustered network, the inference procedure will terminate after one or a few steps. Nevertheless, in case of a poorly chosen *MaxP* value, this criterion can be used to calibrate this parameter.

3.2 Group Schema Creation

As aforementioned, the inference of the interest group schema is achieved gradually by merging the schemas of peers on consecutive steps of the path that the merging procedure follows. In this section we present the algorithm that performs schema merging between two peer schemas.

The goal of the merging procedure is to produce a schema that represents the semantics of the majority of the peers that belong to the respective cluster. Therefore, the

[1] Assuming a path of three peers with schemas S_1, S_2 and S_3 in a clustered overlay, $SS(S_1, S_2) \cdot SS(S'_2, S_3)$ is able to give us a value that is close to the real value of $SS(S_1, S_3)$. The reason is that S_2 is a schema that maps most of the semantics of S_1 and S'_2 is very similar to S_2.

merged schema is neither the intersection nor the union of the schemas of the members of the cluster. Assuming that such a cluster comprises numerous peers, it is straightforward that the intersection of their schemas would be probably empty and their union would be a huge schema with specific semantics being represented multiple times. Thus, we need a merging procedure that keeps in the merged schema the most popular concepts of the respective peer schemas. Yet we aim at inferred group schemas that will be representative of almost all their source peer schemas. Thus, we require a merging procedure that performs high compression before throwing away schema elements (i.e. relations or attributes). Note that a peer can join several groups in order to satisfy its need for information and of course the employment of the group schema for query answering is not mandatory: a peer can still propagate a query from peer-to-peer instead of using a group schema. In any case, our approach is parameterized, such that it can allow schemas with broad or narrow semantics (thus, group schemas can keep or not unpopular concepts).

Finally, we require that the merging procedure is based only on available information on the peers, i.e. it exploits solely the peer schemas and the peer mappings. We remind the reader that we assume that peer mappings are GAV/LAV/GLAV and peer schemas are relational, (i.e. the only internal mappings are foreign key constraints). One mapping is considered to be a set of 1-1 correspondences between attributes that hold with an optional set of value constraints on some attributes. Moreover, peers do not carry semantic information about their schemas and mappings.

The schema merging procedure is dictated by the following dictations:

1. Fewer relations with more attributes are preferred to more relations with fewer attributes
2. The semantic relevance of two relations is proportional to the number of correspondences between their sets of attributes
3. If the keys of two relations are mapped thoroughly, both relations are considered to be projections of the same relation with the same key
4. The key of a merged relation consists of the keys of both relations that are merged
5. If two attributes are merged and at least one of them is a key, then the merged attribute is part of the key of the merged relation
6. Correspondences that involve the same attribute imply that all involved attributes are semantically equivalent
7. Correspondences that are based on any value constraints are considered valid only under conditions and do never produce merged attributes.
8. There is a pre-specified constant that represents the maximum number of relations that the schema of the interest group is allowed to have

Dictation (1) is based on the rationale that more relations result in more join operations in query expressions; thus, mapping a peer schema on the interest group schema would be more complicated in this case. Of course, this may lead to relations with too many attributes that are not semantically closely related. However, this is not a problem, since the group schema is not intended for data storage but just for mediation. [2]

[2] In case that the group schema should be populated with data, many semantically irrelevant attributes of a relation would result in sparse tuples with many NULL values.

In dictation (2) a relation is considered as a set of concepts. Since a correspondence between two concepts entails their semantic equivalence, it actually states the obvious fact that two sets of concepts are considered semantically relevant according to the semantic relevance of their members.

Moreover, the attributes that constitute the key of a relation are considered to represent vital concepts of the respective set, i.e. concepts that deterministically characterize the whole set. Dictation (3) states that the key value prescribes univocally the values of the rest of the attributes; thus, if there are two relations with the same key, a value assignment to the set of attributes that are not part of the key of one relation corresponds to one at most value assignment to the respective set of attributes of the other relation. Thus, the two relations can produce one relation with the same key that contains all the remaining attributes from both relations. Actually, dictation (3) is like an extension of dictation (2) that specifies very big weights for the key attributes of relations.

Furthermore, the merge of two relations produces a relation that is actually the set of all involved concepts. Since we assume that each relation is a set of concepts characterized uniquely by a subset of them, the merged set should be characterized by the union of these subsets (dictation (4)).

Dictation (5) ensues from the previous rationale and adds that, if two attributes are merged during the merge of two relations, then if one or both of them are keys, the merged attribute should be a part of the key of the merged relation. The reason is that if two relations are decided to be merged, it means that they are considered as subsets of a greater set of concepts. Dictation (4) requires that this set of concepts is characterized by the the union of the respective keys. If an attribute which is part of a key is merged to produce a non-key attribute, dictation (4) is violated. Therefore, merged attributes should inherit the key property from either of their merged parts.

Dictation (6) reminds us of the assumption that a correspondence is semantically interpreted as an equivalence of the two involved concepts and that equivalence is transitive. In coherence with this, attributes are merged if they correspond to a common attribute. This means that even attributes of the same relation can be merged eventually (see following example).

Since correspondences between attributes are actually parts of mappings between schemas, the first hold under the constraints of the latter. As discussed in [24], joins are assumed to be associative constraints, whereas value constraints are not. Dictation (7) states that value constraints are intentional conditions under which the mapping correspondences hold. Thereupon, these correspondences cannot produce merged attributes, since the existence of the latter in the merged schema is unconditional.

Dictation (8) ensures the satisfaction of dictation (1) but also guarantees the creation of a schema with more than one relations. Actually, the limits set by dictation (8) express the initial requirement for the approximate size of the inferred schema in terms of involved relations and attributes. Since no other semantic or meta-information is provided, the limits are “safety valves” that stop the enforcement of the merging of relations and attributes and ensure the creation of a schema with more than one relations and relations with more than one attribute.

```
Schema Merging Algorithm
Input: the merged schema $S_{IG}$; the peer schema $S_P$ and a set of mappings $\mathcal{M}$ between them; a set of intra-schema mappings $\mathcal{M}_i$ and a dictionary $D$.
Output: the new merged schema $S'_{IG}$, a set of intra-schema mappings $\mathcal{M}'_i$ and a dictionary $D'$.
Step1: Add to $S_{IG}$ all the relations of $S_P$.
Step2: If $\mathcal{M} = \emptyset$ set $S'_{IG} = S_{IG}$ and go to step 7.
Step3: Merge relations that share the same key using the Relation Merging Procedure.
Step4: While the number of relations is over the limit do:
    a. Select pairs of relations that have the most correspondences between their attributes and that do not depend on value constraints.
    b. From pairs of (a) select the pairs of relations that have the fewest not mapped attributes and merge them using the Relation Merging Procedure.
    c. Remove from $\mathcal{M}$ the mappings used for the merge of (b) and add the involved correspondences in the dictionary $D$.
Step5: Set $S'_{IG} = S_{IG}$, $\mathcal{M}'_i = \mathcal{M}_i \cup \mathcal{M}$.
Step6:Return $S'_{IG}$,$\mathcal{M}'_i$, $D'$.
```

Fig. 4. Schema merging algorithm

The schema merging algorithm is presented in Figure 4. Steps 3 and 4 of refer to the merging of a pair of relations. The procedure shown in Figure 5 performs the merging of two relations according to the following definition.

Definition 4 (Merge of two relations). *The merge of two relations* $R_1(A_1,...,A_n)$, $R_2(B_1,...,B_m)$ *is a relation* R_1/R_2 *with attributes the set* $(A_1,..,A_n) \cup (B_1,...,B_m) - (A_1,..,A_n) \cap (B_1,...,B_m)$

At the end of the schema merging procedure, i.e. when all relevant peer schemas have been merged, relations and relation attributes that have been met very rarely during the procedure can be dropped.

As shown in Figure 4, the schema merging algorithm produces the interest group schema, S_{IG}, but also a set of peer mappings, $\mathcal{M}$, a set of internal mappings, $\mathcal{M}_i$ and a dictionary, D. When the schema merging algorithm is initialized, all four parameters are empty. Each time the schema merging procedure is propagated to another peer, P, S_{IG} is augmented with the relations of P and $\mathcal{M}$ becomes the set of mappings that P maintains with the current form of S_{IG}. We remind that in each iteration the merging algorithm is propagated to the peer p that corresponds to the first element of the stack ST (see Section 3.1). The mappings $\mathcal{M}$ are actually the mappings of schema S_P with the schema of a peer that has already participated in the group inference procedure. The internal mappings are the peer mappings that were not consumed in the successive schema merges. The internal mappings hold additional syntactic and implicit semantic information for the interest group schema elements; thus, they can be very helpful to peers that would like to join the group and create mappings to their local schema. More-

Relation Merging Procedure
Input: A pair of relations R_1, R_2 a set of mappings $\mathcal{M}$.
Output: The merged relation R.
Initialization: $R = \emptyset$.
Step1: Add to R all attributes of the relations R_1, R_2.
Step2: Until the number of attributes is above the limit, if it is possible do:
a. if there are any, merge attributes that are involved only in one correspondence: e.g. correspondence $\{a = b\}$ produces the attribute a/b; if either a or b is a key, make a/b a key.
b. merge the attributes that are involved in at least one correspondence, starting from those participating in the fewest correspondences; in this case produce one attribute for all correspondence: e.g. correspondences $\{a = b\}$ and $\{a = c\}$ produce the attributes $a/b/c$; if either a or b or c is a key, make $a/b/c$ a key.

Fig. 5. Relation merging procedure

over, this set of mappings has the collection of all mappings with value constraints met during the merging procedure. This kind of mappings cannot be consumed: the involved relations/attributes cannot be merged, since they are mapped under certain conditions (the value constraints).

Furthermore, the merged schema has alternative keywords for the same element that result from the merged mapping correspondences. These alternatives are entered in the dictionary D that accompanies the group schema; thus, D is a set of concept correspondences. Later, when the group schema is available in the overlay, the dictionary can be proved of great help for the peers that want to become members of the group.

Step 4 of the algorithm merges relations that do not share the same key. Priority is given to relations that share most of their attributes. Additional criteria in order to break ties can be based on whether the corresponded attributes are parts of the relation keys, or whether unmapped attributes are parts of the relation keys. Nevertheless, refining the algorithm based on additional criteria is out of the scope of this work.

Example Assume that Dr Davis is a doctor that owns a peer database, the schema of which is:
$S_{DavisDB}$:
Visits(Pid, Date, Did)
Disease (Did, DisDescr, Symptom)
Treatment (Did, Drug, Dosology)

And Dr Lu is another doctor with a peer database, the schema of which is:
S_{LuDB} :
Sickness(Did, AvgFever, Drug)
Patients(Insurance$\sharp$, Did, Age, Ache)

The schemas of DavisDB and LuDB are presented in Figure 6; the databases have the following mapping:

$M1_{DavisDB\_LuDB}$:
Disease (Did, _, Symptom), Treatment (Did, Drug, _):-Sickness(Did, AvgFever, Drug), {Symptom = AvgFever, Disease = Sickness}

where the correspondences Symptom = AvgFever, Disease = Sickness that are implied are added in a set at the end of the mapping. [3]

In this case, as shown in Figure 7 there are three correspondences that are encapsulated in mapping $M1$. We assume that the peer of Dr Davis initializes the schema merge. Thus, S_{IG} is initialized to $S_{DavisDB}$. When the group inference procedure iterates on LuDB, after the 1^{st} step of the schema merging algorithm, S_{IG} contains all the relations of $S_{DavisDB}$ and S_{LuDB}. Since there is a mapping among the relations, Step2 is skipped. The algorithm goes on to Step3: relations *Disease* and *Sickness* are merged in one, since they share the same key. Thus, attributes *Symptom* and *AvgFever* are merged. The correspondence $Disease/Sickness.Drug = Treatment.Drug$ is kept as an internal one (Step5). Also, the dictionary D is enriched with correspondences $Disease = Sickness$ and $Symptom = AvgFever$ (Step4c); actually the schema keeps one name for each relation or attribute from the alternative ones. At the end of the schema merging procedure we propose that the schema keeps for relation and attribute names the most common ones encountered during the procedure.

Assuming that the algorithm goes on to Step3, relations $Disease/Sickness$ and *Treatment* are merged, since they are the only ones related with a mapping. Figure 9 shows Step3 of the algorithm. Now there is one attribute named 'Drug' and it is part of the relation key, even though just one of the attributes that were merged was a key. Additional iterations can merge relations based on foreign key constraints, since no other internal mappings exist.

At this point we revise the example based on the assumption that the initial peer schemas $S_{DavisDB}$ and S_{LuDB} have mapping $M1$ but also the following mapping:

$M2_{DavisDB\_LuDB}$:
Visits(Pid, _, Did), Disease (Did, _, Symptom), Treatment (Did, Drug, _):-Sickness(Did, _, Drug), Patients(Insurance♯, Did, _, Ache),{Pid = Insurance♯, Symptom = Ache and Disease = Sickness}

Figure 10 shows almost all correspondences of mappings $M1$ and $M2$. Again, at Step3 of the merging algorithm the relations *Disease* and *Sickness* are merged, since they share the same key. The first iteration of Step4 merges relations *Visits* and *Patients*.

[3] The mapping is actually a view defined on DavisDB.Disease joined with Davis.Treatment, which is matched with relation LuDB.Sickness, such as:
$View_1$(Did, Symptom, Drug):- Disease(Did, DisDescr, Symptom)Treatment (Did, Drug, Dosology)
$View_2$(Did, AvgFever, Drug):-Sickness(Did, AvgFever, Drug)
The mapping is actually:
$View_1$(x, y, z):-$View_2$(x,y,z)
For simplicity, we summarize mappings by omitting view definitions, adding a set with the implied correspondences in the mapping, and introducing '_' for attributes that are not interesting to the mapping.

The state of merging is presented in Figure 11. The second iteration of Step4 merged relations $Disease/Sickness$ and $Treatment$, rather than $Disease/Sickness$ and $Visits/Patients$: even though these two pairs have the same amount of correspondences, the second pair has more unmapped attributes. The result is shown in Figure 12. Finally, assuming that Step4 iterates one more time. All relations of the initial schemas are merged in one, the key of which is the set of all key attributes of the initial relations. Moreover, this step produces an attribute $Symptom/AvgFever/Ache$ where both correspondences Symptom = Ache and Disease = Sickness are merged.

Limitations of the Schema Merging Algorithm

It is apparent from the previous example that diverse concepts may be merged because they correspond to the same concept. Additionally, in this case the dictionary will have entries that do not seem semantically correct at first glance. The example shows that the concepts $Symptom$, *AvgFever* and *Ache* will be merged in one, producing the dictionary entry AvgFever = Ache. The latter equates the concepts *AvgFever* and *Ache*; yet the average person knows that these concepts do not have the same meaning. This problem arises from dictation (6) that is based on our initial assumption/definition that a correspondence indicates the semantic equivalence of the involved concepts. Albeit, the real meaning of concepts *AvgFever* and *Ache* is actually a specialization of the semantics of $Symptom$. An ontology could represent this situation as the hierarchy in Figure 14. Yet our basic assumption is that peers do not have any semantic information about their schemas and mappings other than the semantics that can be deduced from the structure of the schema and the mappings itself. Dictation (6) is coherent with this assumption and it draws the best possible semantic conclusions by collapsing a possible hierarchy of concepts to a simple set of equivalent concepts.

Using additional semantic information such as ontologies and associative tools such as operations on ontologies in order to refine the schema merging algorithm is out of the scope of this work. However, it is possible for peers to use whatever means in their disposition in order to perform a finer schema merging.

Another limitation of the schema merging algorithm is the inability to selectively merge attributes that are corresponded under value conditions. Assume that *DavisDB* and *LuDB* have the following mapping:

$M3_{LuDB\_DavisDB}$:
Visits(Pid, _, Did), Disease (Did, _, Symptom), Treatment (Did, Drug, _):-Sickness(Did, _, Drug), Patients(Insurance♯, Did, Age, Ache), Age $>$ 13

As discussed in [24, 25] the value constraint Age $>$ 13 may have two reasons of existence: the first is that the two databases want to exchange data only for patients that are under 13, but they could exchange data for patients of all ages; the second is that one of the databases stores data only for patients under 13, simply because the owner of the database is a pediatrician; in this case the databases could not exchange mutually data on patients of all ages. Furthermore, in the second case the relations of the pediatrician's database are semantically bounded to the notion of children, as the mapping indicates. This means that 'sickness' is not another word for 'disease' but for 'children's disease'.

In the first case, where the value condition is actually an agreement on which portion of data peers will exchange information, semantic schema merging could and probably should ignore the constraint and merge the respective relations. Yet, in the second case, we have the same situation as with the multiple correspondences of one attribute, discussed above. This means that the value constraint actually indicates that one or more relations of one database, (the database without constraints), involved in the mapping are semantically specialized concepts of relations of the other database, (the one on which the constraint is defined); thus, 'sickness' (or 'children disease') can be a specialization of 'disease'[4].

Certainly, we can follow the same tactic as for multiple correspondences on one attribute, and merge relations mapped under constraints, ignoring the latter. However, we choose not to do this. The reason is that vey often relational schemas are designed such that some relations are specializations, in a semantic way, of another, and the general and specialized relations are joined with foreign keys. For example a database of a hospital can have a relation Person(ID, Role, ...) where attribute Role has values such as 'Doctor', 'Nurse', 'Patient' etc. Also, the database has a relation Doctor(ID, Name, Specialty, ..), where Person.ID is a foreign key in relation 'Doctor'. It is more likely that we do not want to merge these two relations, because we do not want all persons working in the hospital to correspond semantically to the same schema element.

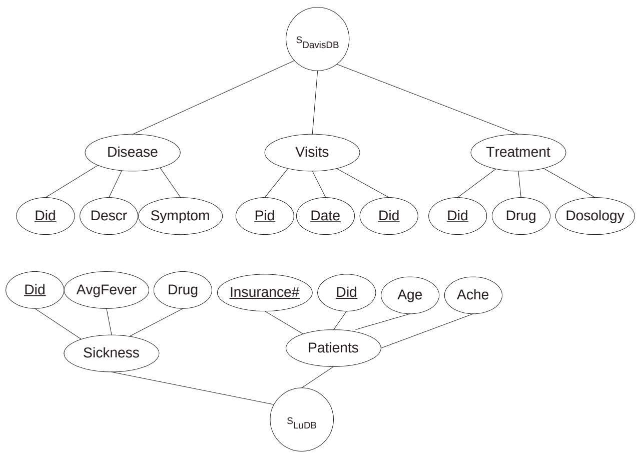

Fig. 6. Two schemas to be semantically merged

[4] We believe that this semantic situation does not occur with value constraints from both databases involved in a mapping. In case of a mapping with constraints on both databases, then concepts from one database overlap semantically with concepts of the other, under certain circumstances.

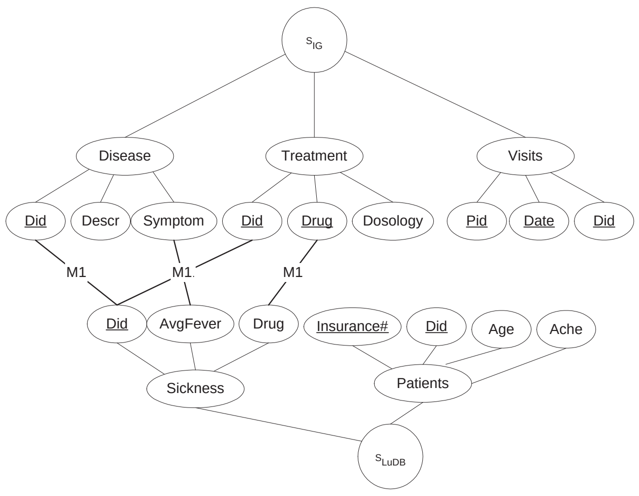

Fig. 7. S_{IG} is initialized to $S_{DavisDB}$ and there is mapping $M1$ between S_{IG} and S_{LuDB}

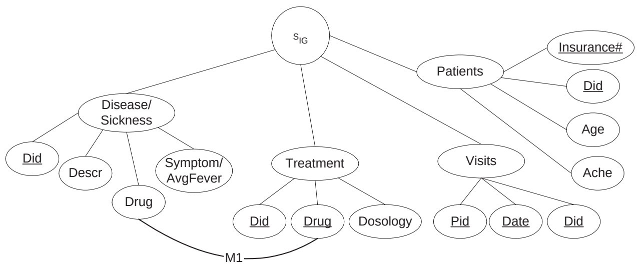

Fig. 8. Relations Disease and Sickness of Figure 7 are merged

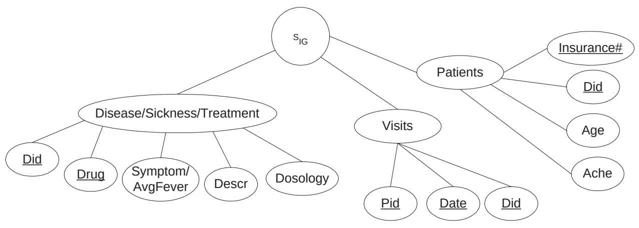

Fig. 9. Relations Disease/Sickness and Treatment of Figure 8 are merged

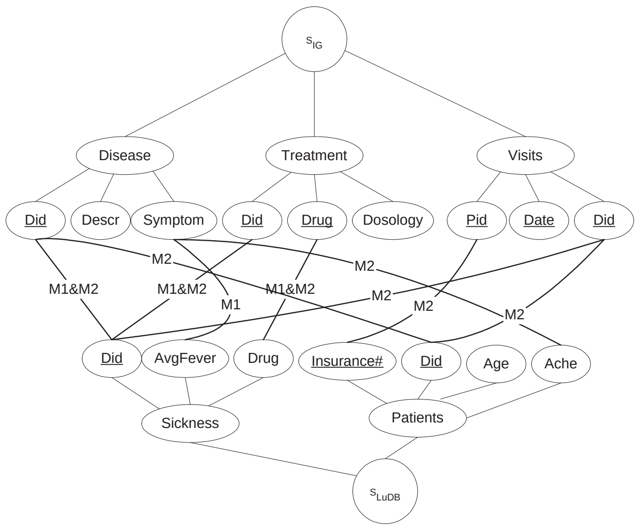

Fig. 10. S_{IG} is initialized to $S_{DavisDB}$ and there are two mapping $M1$, $M2$ between S_{IG} and S_{LuDB}. The most important correspondences are shown

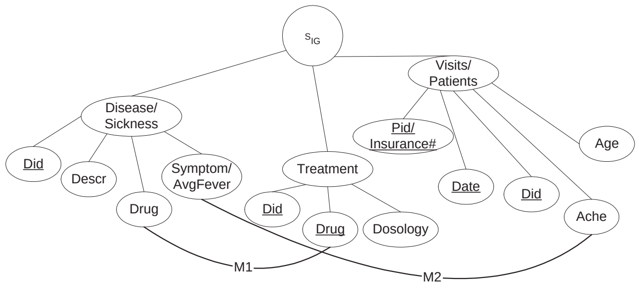

Fig. 11. Pairs of relations Disease and Sickness, and Visits and Patients of Figure 10 are merged

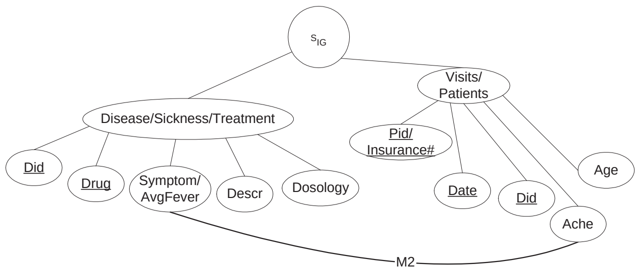

Fig. 12. Relations Disease/Sickness and Treatment of Figure 11 are merged

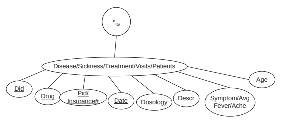

Fig. 13. Relations Disease/Sickness/Treatment and Visits/Patients of Figure 12 are merged

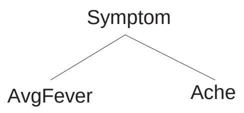

Fig. 14. The hierarchy of concepts *Symptom*, *AvgFever* and *Ache*

3.3 Group Schema Broadcast

The group schema creation is followed by the periodical propagation of respective metadata. These metadata include the group schema, some or all of the IDs of participating nodes (*contact list*), the time of creation and the originator. The reason that a peer may want to know explicitly about peers of the same group is that:

a. it may want to be acquainted with some of them that have very similar schemas to its own, so that it can send queries directly to them (direct rewritings may be more information preserving that going through the group schema)
b. if queries are not broadcasted in the overlay (broadcasting will overload the network), even if they are rewritten to the group schema, they will not go far into the system due to the constraint TTL; thus, they may not reach many or at all peers that are members of the respective group.

Of course, if the clustering procedure performs long before the group schema creation, peers of the same group will already be acquainted, and thus, close to each other, eliminating the (a) and (b) problematic situations. Yet, peers may come and go or even change their schemas. Thus, the achieved clustering is unstable encountering the dynamic nature of the P2P system.

Any peer can rewrite its queries to the group schemas available. Queries can then be directly forwarded to the group members. In this way, we manage to bypass the information loss of multiple rewritings, since a query is translated only once, through the group schema. Making the participating nodes known to all peers enables any remote node to enter the cluster. Furthermore, group nodes have complete mappings with the inferred schema, so no loss is observed there. Peers can now become acquainted with group nodes that have very similar schemas with them, without having to wait to be gradually clustered.

3.4 Group Inference Interaction

While our completely decentralized approach in group creation is necessary, it also raises some consistency issues, since more than one group can be created, even simultaneously. This can affect correct behavior only if nodes similar to the initiator choose to create a group *and* the two processes overlap in the overlay. If two or more peers initiate the group inference procedure, it may be the case that the inferred schemas overlap semantically. This is the case when the schemas of the originators are more similar than the sum of the *radiuses* of the respective group schemas (see Figure 15). In these cases a big number of groups can be inferred that overlap significantly. This is not desired, because peers are disorientated, having many similar choices for groups to participate in, and the network and individual peer overhead for group maintainance is too big. Topologically close peers initiating the process over different semantic groups pose no problem. The same is true if the initiators' hop distance is such that would not allow either procedure to incorporate both groups in its progress. In the following we discuss the details of such situations in order to determine the directions of a protocol for the resolution of group inference interactions, in order to avoid the inference of unnecessary groups.

If two peers close in the overlay initiate the schema inference procedure, it may be the case that their semantic distance is smaller than the radius of one (or both) of the inferred group schemas (see Figure 15 (a) and (b)). In order to avoid extended negotiation rounds between competing potent originators, we require that initiators announce their intention to create a group to their neighborhood. This forces competing initiators with schemas similar to the first initiator to postpone or abort their process, if they are inside the announcement neighborhood. We note that the announcement neighborhood must have a radius proportional to the semantic radius of the group to be inferred. If such peers do not eventually participate in the group inference, they can add themselves to the overlay neighborhood or participate in the consequent *maintenance* of the group.

The originator should announce, i.e. broadcast, the initiation of the group either to the whole overlay (which is definitely inefficient) or to a certain TTL distance, D. The value of D is determined by the intented semantic width of the group being inferred (denoted by the semantic similarity threshold of the group, t, and implied by the threshold on the maximum participating nodes, $MaxP$), as well as the clustering degree of the respective cluster, Cd, (i.e. the quality of the cluster at the point of the respective group initiation). Thus, $D(g) = fun(t(g), MaxP(g), Cd(g))$, where fun is a function.

Optimization Choices for the Group Inference Interaction Protocol

Instead of requiring peers residing in a group announcement area to postpone or abort their intention for group creation, we have the following options:

1. Peers in D reply to the initiator of the group that they want to participate as additional originators of the group; the group initiator initiates the schema merging procedure for a group with multiple origins (i.e. source schemas).
2. Peers in D have the chance to add their selves as originators to a group being inferred in which they already participate, only when the procedure of the latter is iterated on their platform.

The above options intend to be more gentle to other peers that desire to create a group, and give them the chance to be additional originators to a group that has already been announced. The simplest choice, case (1), is to let the peers that learn about the initiation of a group to reply to the initiator that they wish to become *additional originators*.

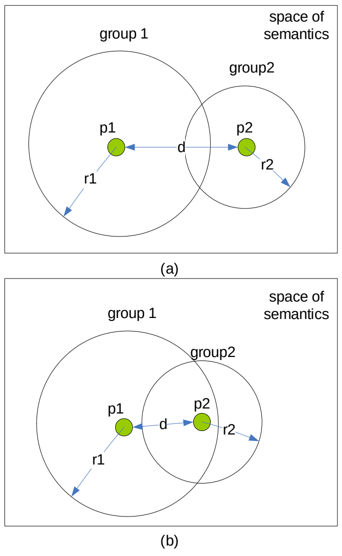

Fig. 15. Semantic overlap of group schemas

The initiator should wait for a small period after the group announcement in order to receive such replies. Then, it begins the procedure taking into consideration the schemas of the additional originators. In a more dynamic environment the protocol can allow the spontaneous participation of peers as additional originators to an ongoing group inference procedure, choice (2). Yet, this is only possible at the time when the inference procedure is iterating on such a peer. At that time, the peer can decide if it wants to introduce another origin of the group schema being inferred. The new origin is its own schema: thus, the group schema will have one more originator from this peer and on. Of course, the sequence of originators that participate in the inference of a group schema has a role in the resulted group schema, especially if it has a small maximum size and if the maximum number of participants for a group is small.

In both cases (1) and (2) the protocol should prescribe a constraint for the maximum number of additional originators and the maximum semantic similarity and dissimilarity between the pioneering originator and the rest. Otherwise, if too dissimilar originators are added the group may become too broad and distort from the interests of the first originator. Moreover, if there is a big dissimilarity among originators and the maximum number of participating peers is small, the merged schema may summarize divergent concepts. Also, if too similar or too many similar originators to the initiator are added, then the inference procedure is overloaded with many unnecessary iterations (the stack ST may become too big). Hence, in both cases (1) and (2) there should be a maximum number of additional originators $MaxO$, which should be significantly smaller than the maximum number of participating peers: $MaxO \ll MaxP$. Moreover, additional originators O should be in a constrained semantic distance from the initiator: $l < SS(S_I, S_O) < u$, where l, u are the lower and upper borders of the semantic distance.

3.5 Group Schema Merging

Nevertheless, peers are eligible to initiate a new group if they have not received a relevant announcement or if they incorrectly calculate their similarity with a known initiator. Possibly such originators will create groups that have a significant semantic overlap with existing ones. Thus, these groups are subject to be merged into a unified schema. The merging of the two group schemas can be performed with the schema merging algorithm. In such a case, the merging will be guided by the mappings that exist between the group schemas and participating peers that are common in the two groups. After both groups are advertised, the respective originators can detect the similarity between the inferred schemas and initiate the merging process. This involves choosing a new originator among the two existing ones, merging the two schemas and advertising the new group using the new initiator and the union of the contact lists.

If an initiating inference procedure is not announced or is announced to a certain *D*, then the only way for a peer beyond *D* to have knowledge about an ongoing inference procedure is to have already participated in it. Peers that have not yet participated in an ongoing procedure [5], are potent of initiating a new one, even if it turns out later that they are part of a procedure initiated earlier. The parameter that determines the interaction of a pair of inferred groups is their semantic overlap. If groups do not overlap, then there is no interaction, and, thus, no problem to solve. If groups overlap, either:

– none of the group originators belong semantically to the group of the other.
– one of the group originators belongs semantically to the group of the other.
– both of the group originators belong semantically to the group of the other.

The above cases provide a reason to merge the respective groups. If both originators belong to each other's group, then there is a strong reason to merge the groups. If one or none originator belongs to the other's groups then there is a weaker reason to merge the group. Nonetheless, the semantic overlap of two groups is calculated with the *SS* function, presented earlier. The decision for group merging aims to better query answers overall. We think that this will be the case for peers that belong marginally to the groups, thus, they do not get satisfying answers through them.

From the above discussion we can conclude that interaction and overlap among groups that are or have been inferred is inevitable. Thus, the P2P overlay has to guarantee that group merging decisions are based solely on the group schema semantics and not on the execution of the inference process, i.e. the group initiator and additional originators.

An important property must hold:
Groups created by similar initiators will also be similar and groups by dissimilar initiators will be dissimilar.

This property is essential because it justifies that authority peers can independently initiate the process (and thus block other similar ones from doing it). The semantic clustering of the peers that belong to a social network assures that this property holds. Moreover, this property shows that groups are characterized by their schema and their

[5] We assume that the broadcast of new groups is short enough, so that the period between the end of the procedure and the end of the broadcast is negligible.

inference process does not matter.

Finally, we note that group schemas are not connected to each other unless this becomes evident as soon as they are created and is resolved through the merging procedure. As mentioned before, peers select the group of their choice to send the query to (and they are not limited to choosing a single one). If posed queries and employed groups are not semantically very relevant, then the querying peers may not be satisfied with the answers they get through the groups. In this case, the can always propagate their queries in the network without using the groups. Through the employment of the merging procedure, the presence of many very similar groups is avoided, and , thus, peers can choose groups without confusion.

3.6 Group Schema Maintenance

A group schema has to be occasionally maintained. The group schema maintenance includes maintenance of:

a. the point of contacts, since new peers that may belong to the group join and old peers that belonged to the group leave.
b. the group schema itself, in order to represent the peer schemas that are members of the group (since old peers leave and new join).

There are two ways to decide when to maintain a group schema:

1. maintenance is performed periodically
2. maintenance is performed when the quality of answers to queries rewritten to the group schema by members of the group is not satisfying.

The maintenance process refers to updates in the contact list as well as the group schema itself. Maintenance is necessary, since peers join the group while others that belong to the group leave or change their local databases in time. There are two ways to decide how to maintain a group schema: The first is to allow the originator to initiate the inference process periodically. The second is to allow *any* eligible peer re-start the process. In order for both approaches to work, we define an *epoch* factor to represent the maximum life-span of a group, after which it will become invalid. Then, the originator can invoke the inference process every *epoch* minutes and re-transmit the new group in the overlay. Thus, group meta-data are kept in a form of soft state and get promptly updated. By allowing any eligible peer to undertake the role of the originator, we eliminate inconsistencies created by changes in the original initiator and also ensure that the inferred schema does not specialize.

Obviously, there is a trade-off between the cost of repeating the process over the anticipated query performance using stale groups. However, PDMSs are considered as P2P applications that do not exhibit very dynamic changes in connectivity or peer interests (i.e. peer schemas). Rather than this, peers are expected to stay connected (and therefore have mappings) to the same peers and store the same kind of data for long periods. In any case, peers can be parts of a group, answer and route queries without having to pose queries at the same time (i.e., without active participation from the user).

In the future we intend to exploit the performance of maintenance at time points when the quality of answers to queries rewritten to the group schema by members of the group is not satisfying.

3.7 Group Deletion

Up to this point we have discussed all issues that concern the creation and life of a semantic group. However, after a group has been around in the P2P overlay for a while it might have to be deleted altogether. A recommendation for a group deletion can be based on several reasons. First, a group may not be used anymore for query answering or for meeting acquaintances. Second, after using a group for some time it may turn out that it is problematic: the group schema may have flaws due to errors during the performance of the inference procedure. Third, participants of a group may decide that they do not want to be members of it anymore. In this case, if too many participants drop out, it may be more meaningful to delete the group and let peers initiate new group inference procedures, than to maintain the group.

In any case, obsolete peer needs or procedural mistakes should not remain perpetually in the overlay. Due to all these conditions, group deletion should be a permitted operation that complements the dynamic creation and maintenance of groups. Yet, it is not easy to delegate the decision for group deletion. As such, group deletion is implicitly handled through the periodic group maintenance process. If group deletion is necessary due to nodes changing their interests (i.e., local schemas and queries), an extreme case of which is to have departed from the system, at the next maintenance check point the necessary conditions for an initiator will not be met (i.e., the number of posed and answered queries over a specific accuracy threshold), invalidating all current and past group information without any extra coordination between its participants. We prefer this solution to an explicit one that would require intra-group communication since it integrates with the maintenance operation and requires no manual intervention from the part of the users.

4 Performance Evaluation

To evaluate the performance of the proposed group inference procedure, we use a message-level simulator that implements it over an unstructured overlay of semantically clustered nodes. The clustering is performed using the *GrouPeer* system [25].

Overview of *GrouPeer*
GrouPeer focuses on the problem that, in a random flat unstructured peer-database system, information-rich peers may well remain hidden to query initiators because of the enforced reformulation of queries on each node of the propagation path. It proposes a procedure that supports the evasion of successive rewritings on every peer of a query's propagation path, instead of, sometimes hopelessly, refining query reformulation. This methodology enables peers to discover others with similar interests and schemas, that cannot be tracked otherwise. Pairs of remote peers that exchange queries and answers learn gradually about the schema of the other party. Learning is performed through

making queries and evaluating their answers, and is formed in mappings between the schemas of the two peers.

In GrouPeer, peers decide to add new (and abolish old) one-hop neighbors in the overlay (acquaintees) according to the accuracy of the answers they receive from remote peers. This is measured using a function that tries to capture the semantic similarity between rewritten versions of a query. Specifically, requesters (i.e., peers that pose queries) accumulate correct and erroneous mappings with remote peers through a learning procedure. Based on these mappings, they decide to become acquainted with peers that store information similar to their interests. The result is an effective semantic clustering of the overlay, where the accuracy of query rewritings and answers is a lot higher compared to the unclustered overlay (for details see [25]).

We compare the query evaluation performed by GrouPeer with the evaluation that utilizes the inferred groups on the overlay. In GrouPeer a query is propagated in the P2P network using informed walkers. A query is successively rewritten on each peer on a query propagation path. The rewritten query on each peer is answered using the local schema and data. This procedure of query answering is the state-of-the-art in flat PDMSs, [1, 3, 17]. When the first group is created, we direct relevant queries to the inferred schema. The basic performance metrics are the average *accuracy* of answers to the original queries (i.e., the similarity of the rewritten query that is answered over the original one), as well as the number of nodes that provide an answer. Similarity is calculated by a formula presented in [25] that identifies erroneous or not-preserved correspondences in mappings, which degrade the complete and perfect rewriting. Semantic query similarity is a very big issue, [24], and is out of the scope of this paper. Very briefly, the essence of the query similarity formula that is used in GrouPeer and that we use in this experimental study is:

$$M_{sim}(Q_{orig}, Q_{ans}) = \frac{\sum Q_{orig}\ elements\ present\ in\ Q_{ans} + \sum additional\ Q_{ans}\ elements}{\sum Q_{orig}\ elements} \tag{2}$$

In function (2) *elements* are either query attributes or query conditions (for example, for an SQL (select-project-join) query, elements are the 'select' clause attributes and the 'where' clause conditions). To identify the gains of our grouping approach, we present the percentile increase/decrease in accuracy and number of answers compared to GrouPeer's clustering as these are measured on the *first* created group. Participants of the group hold mappings with the group schema; thus, when the query is rewritten to the group schema, the successive rewritings through the chain of mappings are avoided. In the presence of a group, there is a query rewriting from the peer schema on which it is originally posed to the group schema, and a second rewriting from the group schema to each peer that will answer the query. Data is stored locally in each peer and they are not re-materialized in the group schema. Non-members create mappings with relevant group schemas.

We present results for 1,000-node random graphs (an adequate number of participants regarding our motivating application) with average node degrees around 4, created by the *BRITE* [29] topology generator. Results are averaged over 20 graphs of the same

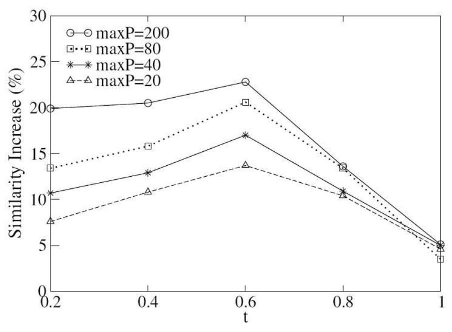

Fig. 16. % Increase in answer similarity over variable MaxP and t

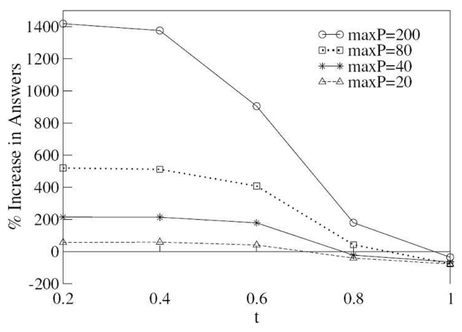

Fig. 17. % Increase in number of answers over variable MaxP and t

type and size, with 100 runs in each. Results using power-law topologies constructed by Inet-3.0 [20] with the same number of peers are qualitatively similar.

For the schemas stored at each node, as it is realistically very hard to come up with hundreds of different but yet semantically similar ones for each node, we use two initial relational schemas, whose tables and attributes are uniformly distributed at nodes. The initial schema comprises of 5 tables and 33 attributes. Seven attributes are keys with a total of 11 mappings (correspondences) between them. Each peer stores 10 table columns (attributes) on average. Queries are formed on a single or multiple tables if applicable (join queries). We also experimented with a larger schema of 8 tables with a total of 55 attributes that was retrieved from the Internet movie database [19] using *JMDB* [21]. JMDB is a Java-based application to locally search for information about movies, actors, producers, directors, etc as these are provided through IMDb. This second schema comprises of 12 keys and a total of 14 mappings between the attributes.

Our metrics are the percentile increase/decrease in accuracy and number of replies compared to clustering as these are measured on the *first* created group. We use the terms *accuracy* and *similarity* interchangeably. The maximum size of the inferred schema is always in the order of the size of the initial schema used to produce the local ones during start-up. When the first group is created, we direct relevant queries to the inferred schema and measure their similarity compared to the semantic clustering of the social network at the time of group creation. Initiators that belong to the group hold the complete mappings with the group schema, avoiding reformulation errors. Non-members utilize the same learning feature as with normal nodes, assuming a “virtual” host holding the group schema as their contact.

First, we vary the maximum group size limit, *MaxP*, as well as the minimum similarity of participating peers to the initiator node, t. Figures 16 and 17 show the obtained results for 100 requesters and maximum 100 queries each. As t increases, the group becomes more specialized and less general. In contrast, small similarity values produce groups too general that incorporate many concepts foreign to the initiator. Initiators choose to send queries to a schema if they deem it advantageous. This has the effect that *specialized* groups (i.e., high value of t) receive fewer queries, while more “gen-

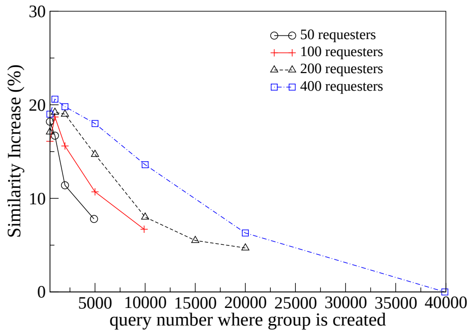

Fig. 18. % Increase in answer similarity over variable group creation time

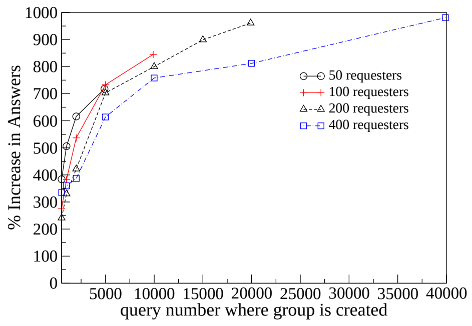

Fig. 19. % Increase in number of answers over variable group creation time

eral" ones receive more but cannot answer them all satisfactorily. Thus, there exists a point where grouping ceases to increase its relative gains to clustering, as our graphs show.

Both metrics increase as *MaxP* increases. This is reasonable since more nodes can participate and produce results. Very specialized grouping causes significantly less populated groups, which in turn affects the number of returned answers. As groups get more general (around $t = 0.6$), an improvement of 13-23% in accuracy is achieved, while the gains in replies are 40-900%. As t decreases, the gains in accuracy decrease but more results are generated. These curves show that a t value of around 0.65 with the group initiator and $MaxP = 80$ achieve good results without too much generalization. These will be our default values for the rest of this discussion.

Next, we try to determine the quality of the created group based on its creation time, i.e., the number of queries at which it was created. Figures 18 and 19 show the percentile improvement in our basic metrics when the first group is created at various points in the clustering process. Our observations show a decrease in the relative gains in accuracy and an increase in the corresponding number of answers. This happens because clustering improves with time while the number of results slightly decreases due to the forwarding process: now more walkers cross paths on relevant nodes. What is important is that groups that are allowed to be created as soon as possible (which would be the frequent case) show about 20% more accurate answers and return about three times more results compared to the clustering of the social network, even though the inference procedure is performed on a less optimally clustered overlay. Groups that are created later exhibit noticeable gains, especially in terms of the number of replies.

Table 1 shows the exact performance figures using our default parameters for 400 requesters and various queries-per-requester combinations for both schemas. The figures in parentheses show the percentile increase compared to simple clustering for the same number of queries. We notice that querying the inferred groups results in an average 18% increase in accuracy and a 300-400% increase in the number of replies. This is true regardless of the number of requesters or their querying rates. It is interesting to note that, in all these results, the queries from nodes inside the created groups are less

Table 1. Performance comparison with clustering

	initial schema		IMDb schema	
qu/requ	**Sim**	**#Answ**	**Sim**	**#Answ**
10	0.70(+19.9%)	53.7 (+387%)	0.68(+16.7%)	50.1 (+317%)
50	0.71(+19.0%)	61.6 (+461%)	0.69(+17.3%)	52.6 (+341%)

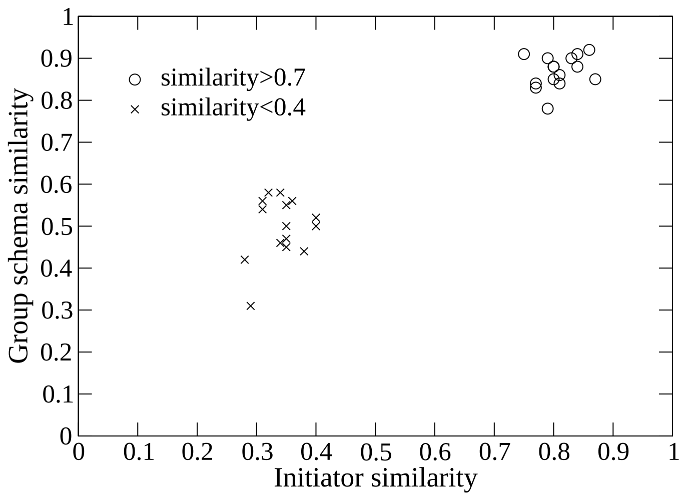

Fig. 20. Relationship between initiator and inferred schema similarity

than 10% of the total. This proves that group creation and propagation effectively helps all nodes in the overlay. The figures regarding the IMDb schema are slightly lower due to the fact that the initial average similarity between peers is lower (for the same number of stored attributes per node as in the initial schema).

One of the basic assumptions of our scheme is that each peer can individually choose to initiate the group inference process. This allows for completely distributed behavior only if semantically close initiators produce similar groups and the opposite. We measure the similarity between the first and randomly selected thereafter initiators as well as of the group schemas created respectively. Figure 20 displays results over different runs, where either the two initiators were over 70% or less than 40% similar. Clearly, for very similar initiators the process yields very similar groups. On the other hand, for fairly dissimilar initial schemas, the created groups are 40-50% similar. This value is a little higher than expected due to the high overlap and semantic relations between stored attributes at various peers. When data is placed in a non-overlapping manner, such groups have less than 20% similarity. So, there clearly exists a correlation between initiator and inferred schema similarity value.

As we just showed, peers with similar schemas generate similar groups. To do so simultaneously is undesirable for two reasons: First, the system will perform a redun-

Table 2. Estimating group broadcast range

	D = 0.5	D = 0.7	D = 0.8	D = 0.9	D = 1.0
Min/Max Distance	1.1/5.9	1.9/5.6	2.1/5.3	2.9/4.8	3.8/4.2
#nodes	597	235	113	27	17
%nodes $<$ 4 hops	78	80	80	78	70

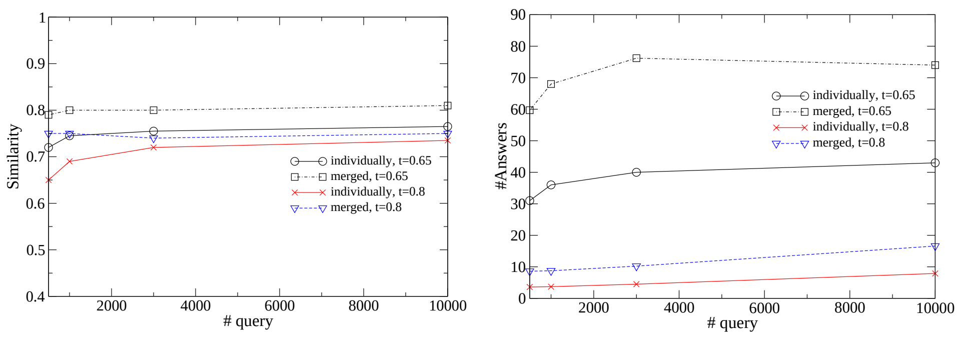

Fig. 21. Similarity and number of answers of the initial and merged groups vs creation time

dant operation and second, it will force our merging process to be invoked regularly. As we mentioned in Section 3, initiators broadcast their intention to create a semantic group. Nevertheless, broadcasts that reach many nodes are very costly. Furthermore, our clustering process assures that a non-negligible number of semantically close nodes will also be close to the initiator in the hop-distance metric. To demonstrate this, we measure the hop-distance distribution of peers not included in the group creation process with similarity greater or equal to D to the initiator, given our default parameters. Table 2 presents our results.

We notice that the minimum distance increases as we search for more similar peers, while the maximum decreases. This is due to the clustering process: Similar peers get closer in the overlay. Grouping includes most of these peers, so the minimum distance to a non-grouped similar node increases. Moreover, the ones that have been left out of the group inference are now closer than before. The results show that a broadcast range of 4 contacts around 80% of our target nodes. Nevertheless, as D increases, these nodes become scarce. Thus, assuming that $D \simeq 0.65$ for practical reasons, a TTL=4 would suffice. In our experiments, a broadcast of that scope blocks an increasing number of nodes with time: after about 6 queries per requester, 115 potential group initiators are blocked on average, while after 20 queries this number increases to over 600. For larger values of D, broadcasting with large range causes the majority of messages to be delivered to dissimilar peers.

Finally, we present some results concerning merging process. When two similar groups are identified (through broadcasting of the group metadata), the merge process is initiated. We measure the similarity and number of replies by the two groups as well as the merged one and present the results in Figure 21. We notice that, while the two groups and the merged one do not substantially differ in the accuracy of the results (although the merged group always outperforms them), the new schema delivers almost twice as many. A very important observation is that the time of creation of the individual groups plays almost no role in their performance, which shows that the social network will keep operating without performance degradation.

5 Related Work

The problem of semantic schema merging is generally related to the problems of schema or ontology matching and integration. The recent survey in [37] approaches in a unified way all these problems, since they are basically dealing with schema-based matching. A survey of ontology mapping techniques is presented in [22]. The authors focus on the current state of the art in ontology matching. They review recent approaches, techniques and tools. Once appropriate mappings between two ontologies have been established, either manually, semi-automatically or automatically, these mappings can be used to merge the two ontologies or to translate elements from one ontology to the other. Examples of tools for ontology merging are OntoMerge [12] and PROMPT [33]. However, creating and maintaining a merged ontology incurs a significant overhead. Moreover, a translation service for OWL ontologies is presented in [30]. The translation relies on a provided mapping between the vocabularies of the two ontologies. Then, a class C_1 from the source ontology can be characterized as *strongly-translatable*, *equivalent*, *identical*, *weakly-translatable* or *approximately-translatable* to a class C_2 from the target ontology, depending on its name mapping and the *translatability* of its associated properties and restrictions.

Schema matching is a fundamental issue in the database field, from database integration and warehousing to the newly proposed P2P data management systems. As discussed in [35], most approaches to this problem are semi-automatic, in that they assume human tuning of parameters and final refinement of the results. This is also the case in some recent P2P data management approaches (e.g., [7, 34]. Generally, schema matching [35] and integration [4] are operations that adhere to schema structure in a strict way. Thus, most of the effort is concentrated in detecting and compromising contradictory dependencies and constraints.

Ontology matching/integration is a very similar problem to schema matching/integration. As discussed in [32, 37], both ontologies and schemas provide a vocabulary of terms with a constrained meaning. Yet ontologies and schemas differ in the declaration of semantics: on one hand ontologies specify strict semantics and on the other hand schemas do not specify any explicit semantics. Because of this vital difference, ontology matching/integration has to follow a strict semantics structure, whereas schema matching/integration has to obey to strict structural semantic-less constraints. Our work in this paper, is an effort to complement these approaches by filling the gap between them. Our focus is the semantics that can be deduced from schemas alone without adhering to the schema structure or to any ontology constraints on semantics.

The Chatty Web [1] considers P2P systems that share semi-structured or structured information. The authors are concerned about the gradual degradation, in terms of syntax and semantics, of a query that is propagated along a network path. However, the

Chatty Web approach considers peers that own very simple relational schemas and GAV mappings with their acquaintees. Instead, we are interested in more complex peer schemas and we consider GAV, LAV or GLAV mappings.

PeerDB [34] facilitates relational data sharing without any schema knowledge. Query matching and rewriting is based on keywords (provided by the users). A two-step process is described: First all nodes within a TTL radius are contacted, returning prospective answer meta-data. Then the user selects the ones that are relevant to the local query and the requester directly contacts the selected sources and asks for the results to the various rewritten versions of the query.

The works in [17] and [23] deal with data exchange between peers. Ref. [17] presents a significant approach to the heterogeneity issue in P2P data management and proposes a language for schema mediation between peers. Also, the authors present an algorithm for query reformulation based on local-as-view as well as global-as-view query answering. In [23], the authors describe mechanisms for the declaration of data exchange policies on-the-fly based on ECA rules. They also propose a general architecture for peer-databases and elaborate on the establishment and abolishment of acquaintances between peers.

Beyond [17] other significant works such as [3, 6, 9, 14, 36] have introduced novel frameworks for PDMSs. All of these works agree to the fundamental principles of peer autonomy, peer heterogeneity and peer data exchange through local pairwise mappings. Our approach complies with these principles and it follows the lines of these works. Nevertheless, the focus of all these works is query answering through propagation from peer-to-peer, whereas our focus is on providing a dynamically adaptable global schema for a semantic group of peers. Therefore, our approach is complementary to these works, since it proposes query answering through a group schema, without annulling the peer-to-peer query propagation. Both of these techniques for query answering can coexist in a PDMS.

Beyond the above significant works, there are plenty that have talked about semantics and semantic clustering of peers. The work in [11] is one of the first to consider semantics in P2P systems and suggest the construction of semantic overlay networks, i.e. SONs. Later on, other researchers have attempted to go beyond the a priori static formulation of SONs: the work in [38] suggests the dynamic construction of the interest-based shortcuts in order for peers to route queries to nodes that are likely more capable of answering them. Inspired by [38], the authors in [42] but also in [18] exploit implicit approaches for discovering semantic proximity based on the history of query answering and the least recently used nodes. In the same spirit the work in [13] presents preliminary results about the clustering of the workload on the real popular systems e-Donkey and Kazaa. Additionally, SQPeer is an extensive work on PDMS that share RDF data and they localize the query patterns using views [26].

Some of the well-known projects that have dealt with the data heterogeneity problem in P2P systems are [2, 16, 31, 40]. Edutella [31] is a schema-based network that holds RDF data. Peers have services (e.g. quering, mapping, mediating etc) that they share with other peers. Peers can formulate complex queries that are translated in wrappers to queries on the Edutella Common Data Model. Peers register their services and the kinds of queries they can answer to mediators. The latter route the incoming queries to peers that are probably able to answer them. Edutella is an interesting effort towards the solution of the heterogeneity problem both of data and services. However, it is not focused on semantic clustering of peers and does not propose sophisticated methods for distributing queries to semantically relevant peers.

Finally, there are some works that employ *Distributed Hash Tables* in order to deal with peer data heterogeneity, such as [2,40], or ontologies, such as [16]. Moreover, most of the works that consider query answering in unstructured networks with super-peers [43] assume the presence of ontologies in order solve the problem of heterogeneity in peer semantics. A variety of such works, [5, 8, 10, 28] assumes that the semantics of peer schemas are described using an ontology model. A super-peer that manages semantically similar peers maintains mappings to these descriptions. A query is then routed to semantically relevant peers following these ontology descriptions. Our work is orthogonal to these works in that we do not assume either the presence of static centralized managing overlay nodes, such as super-peers, or the enhancement of peer semantics with ontology-based descriptions.

6 Summary

In this paper we have described a method to automatically create schemas in order to characterize semantic clusters in PDMSs. Our scheme operates on clustered unstructured P2P overlays. By iteratively merging relevant peer schemas and maintaining only the most frequent common characteristics, we provide a schema representative of the cluster. Group schemas can be used in order to increase both query performance and the volume of returned data. Our experimental evaluations confirm these observations in a detailed comparison with the GrouPeer system.

ACKNOWLEDGMENTS

This work has been funded by the project PENED 2003. The project is cofinanced 75% of public expenditure through EC - European Social Fund, 25% of public expenditure through Ministry of Development - General Secretariat of Research and Technology and through private sector, under measure 8.3 of OPERATIONAL PROGRAMME ”COMPETITIVENESS” in the 3rd Community Support Programme.

References

[1] K. Aberer, P. Cudre-Mauroux, and M. Hauswirth. The Chatty Web: Emergent Semantics Through Gossiping. In *WWW Conference*, 2003.

[2] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. Van Pelt. Gridvine:Building internet-scale semantic overlay networks. In *International Semantic Web Conference*, 2004.

[3] Marcelo Arenas, Vasiliki Kantere, Anastasios Kementsietsidis, Iluju Kiringa, Rene J. Miller, and John Mylopoulos. The hyperion project: from data integration to data coordination. *SIGMOD Record*, 32(3):53–58, 2003.

[4] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[5] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Querying a super-peer in a schema-based super-peer network. In *DBISP2P*, pages 13–25, 2005.

[6] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing : A vision. In *WebDB*, pages 89–94, 2002.

[7] P. A. Bernstein, S. Melnik, and J. E. Churchill. Incremental schema matching. In *VLDB*, pages 1167–1170, 2006.

[8] J. Broekstra, M. Ehrig, P. Haase, A. Kampman F. van Harmelen and, M. Sabou, R. Siebes, S. Staab, H. Stuckenschmidt, and C. Tempich. A metadata model for semantics-based peer-to-peer systems. In *SemPGRID*, 2003.

[9] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *PODS*, pages 241–251, 2004.

[10] S. Castano, A. Ferrara, S. Montanelli, E. Pagani, and G.P. Rossi. Ontology-addressable contents in p2p networks. In *SemPGRID*, 2003.

[11] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. In *Technical Report*, 2003.

[12] D. Dou, D. V. McDermott, and P. Qi. Ontology translation on the semantic web. *J. Data Semantics*, 2:35–57, 2005.

[13] F. Le Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulie. Clustering in Peer-to-Peer File Sharing WorkLoads. In *IPTPS*, 2004.

[14] E. Franconi, G. M. Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *DBISP2P*, pages 64–76, 2003.

[15] Gnutella website: http://gnutella.wego.com.

[16] P. Haase, B. Schnizler, J. Broekstra, M. Ehrig, F. van Harmelen, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *Journal of Web Semantics*, 2005.

[17] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *ICDE*, 2003.

[18] S. Handurukande, A.-M. Kermarrec, F. Le Fessant, and L. Massoulie. Exploiting Semantic Clustering in the eDonkey P2P Network. In *ACM SIGOPS*, 2004.

[19] The internet movie database: http://imdb.com.

[20] C. Jin, Q. Chen, and S. Jamin. Inet: Internet Topology Generator. Technical Report CSE-TR443-00, Department of EECS, University of Michigan, 2000.

[21] Java movie database: http://jmdb.com.

[22] Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: The State of the Art. *Knowl. Eng. Rev.*, 18(1):1–31, 2003.

[23] V. Kantere, I. Kiringa, J. Mylopoulos, A. Kementsientidis, and M. Arenas. Coordinating P2P Databases Using ECA Rules. In *DBISP2P*, 2003.

[24] V. Kantere and T. Sellis. Reusing Classical Query Rewriting in P2P Databases. In *DBISP2P*, 2006.

[25] V. Kantere, D. Tsoumakos, T. Sellis, and N. Roussopoulos. GrouPeer: Dynamic Clustering of P2P Databases. Technical Report TR-2006-4, National Technical University of Athens, 2006. http://www.dbnet.ece.ntua.gr/pubs/uploads/TR-2006-4. Submitted for publication.

[26] G. Kokkinidis, E. Sidirourgos, and V. Christophides. *Semantic Web and Peer-to-Peer*, chapter ”Query Processing in RDF/S-based P2P Database Systems”, pages 59–81. Springer-Verlag, 2006.

[27] A. Y. Levy. Answering Queries Using Views: A Survey. In *VLDB Journal*, 2001.

[28] A. Lo”ser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic overlay clusters within super-peer networks. In *DBISP2P*, pages 33–47, 2003.

[29] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS*, 2001.

[30] L. Mota and L. Botelho. Owl ontology translation for the semantic web. In *Proceedings of the Semantic Computing Workshop of the 14th International World Wide Web Conference*, 2005.

[31] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. Edutella: A p2p networking infrastructure based on rdf. In *WWW*, 2002.

[32] N. Fridman Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[33] N. Fridman Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *AAAI/IAAI*, pages 450–455, 2000.

[34] B. Ooi, Y. Shu, K.L. Tan, and A.Y. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. In *ICDE*, 2003.

[35] E. Rahm and P.Bernstein. A Survey of Approaches to Automatic Schema Matching. In *VLDB Journal*, 2001.

[36] L. Serafini, F. Giunchiglia, J. Mylopoulos, and P. A. Bernstein. Local relational model: A logical formalization of database coordination. In *CONTEXT*, pages 286–299, 2003.

[37] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics IV*, pages 146–171, 2005.

[38] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *INFOCOM*, 2003.

[39] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *SIGCOMM*, 2001.

[40] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *SIGCOMM*, 2003.

[41] I. Tatarinov and A.Halevy. Efficient Query Reformulation in Peer-Data Management Systems. In *SIGMOD*, 2004.

[42] S. Voulgaris, A.-M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting Semanntic Proximity in Peer-to-Peer Content Searching. In *FTDCS*, 2004.

[43] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *ICDE*, pages 49–, 2003.