Summary Management in P2P Systems

Rabab Hayek
Atlas team, INRIA and LINA,
University of Nantes, France
rabab.hayek@univ-nantes.fr

Guillaume Raschia
Atlas team, INRIA and LINA,
University of Nantes, France
guillaume.raschia@univ-nantes.fr

Patrick Valduriez
Atlas team, INRIA and LINA,
University of Nantes, France
Patrick.Valduriez@inria.fr

Noureddine Mouaddib
Atlas team, INRIA and LINA,
University of Nantes, France
noureddine.mouaddib@univ-nantes.fr

ABSTRACT

Sharing huge, massively distributed databases in P2P systems is inherently difficult. As the amount of stored data increases, data localization techniques become no longer sufficient. A practical approach is to rely on compact database summaries rather than raw database records, whose access is costly in large P2P systems. In this paper, we consider summaries that are synthetic, multidimensional views with two main virtues. First, they can be directly queried and used to approximately answer a query without exploring the original data. Second, as semantic indexes, they support locating relevant nodes based on data content. Our main contribution is to define a summary model for P2P systems, and the appropriate algorithms for summary management. Our performance evaluation shows that the cost of query routing is minimized, while incurring a low cost of summary maintenance.

1. INTRODUCTION

Today's information systems are facing two main problems. First, they host a large number of data sources that are highly distributed, autonomous, and dynamic. Second, modern applications generate huge amount of information stored into the connected data sources, which become more and more voluminous. Therefore, these systems need to scale up in terms of the number of participants, as well as in terms of the amount of shared data. In our solution, we propose to combine the two paradigms: P2P and data summarization.

While distributed systems such as database, integration and parallel systems have reached their maturity and only supported a limited number of users, P2P systems allow data sharing on a world wide scale with many advantages like decentralization, self-organization, autonomy, etc. Popular examples of P2P systems, e.g. Gnutella [16] and KaZaa [17], have millions of users sharing petabytes of data over the Internet. However, a major problem in the operation of P2P systems as distributed systems is object locating. Initially, P2P search systems rely on flooding mechanism and its variations. Though simple and robust, this approach suffers from high query execution cost and poor query recall. Many works on P2P systems have addressed the problem of search efficiency, and proposed various techniques that can be classified into four main categories: data indexing (e.g. [1], [30]), data caching (e.g. [4]), mediation (e.g. [21], [28]) and network clustering (e.g. [3]). According to this classification, there are hybrid techniques such as caching data indexes [8], or clustering a P2P network based on index similarity [11].

So far, data localization has been the main issue addressed in P2P systems, since their scalability is constrained by the employment of efficient search techniques. However, nowadays we are asking the following question: with the ever increasing amount of information stored each day into data sources, are these techniques still sufficient to support advanced P2P applications? To illustrate, in a scientific collaborative application, a doctor may require information about patients diagnosed with some disease, without being interested in individual patients records. Besides, a user in today's decision-support applications may prefer an approximate but fast answer, instead of waiting a long time for an exact one. Therefore, reasoning on compact data descriptions that can return approximate answers like "dead Malaria patients are typically *children* and *old*" to queries like "*age* of dead Malaria patients", is much more efficient than retrieving raw records, which may be very costly to access in highly distributed, massive databases.

In this work, we propose a hybrid search technique that consists in building summaries (i.e. *semantic* indexes) over data shared in a clustered P2P network (e.g. superpeer network). Our summaries are synthetic, multidimensional views with two main virtues. First, they support locating relevant nodes based on their data descriptions. Second, they provide an intelligible representation of the underlying data, and allow an *approximate query processing* since a query can be processed entirely in their domain, i.e. outputs are summaries.

This paper makes the following contributions. First, we

EDBT'08, March 25–30, 2008, Nantes, France.

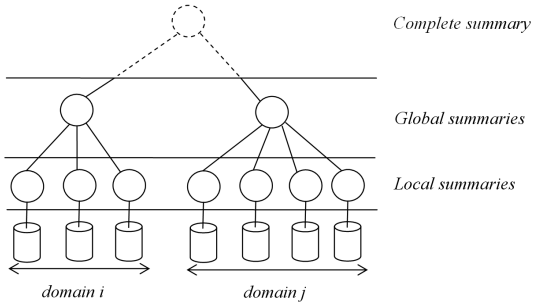

Figure 1: Summary Model Architecture

define an appropriate summary model for P2P systems. We work in the context of superpeer networks, where a *domain* is defined as being the set of a superpeer and its clients. Then, we propose efficient algorithms for managing a summary in a given domain. We validated our algorithmic solutions through simulation, using the BRITE topology generator and SimJava. The performance results show that the cost of query routing, which is measured in terms of the number of exchanged messages, is minimized while incurring a low cost of summary maintenance.

The rest of this paper is organized as follows. Section 2 describes our summary model for P2P systems. In section 3, we present our algorithms for summary management. Section 4 discusses query processing in the context of summaries. Section 5 gives a performance evaluation with a cost model and a simulation model. Section 6 compares our solution with related work. Section 7 concludes.

2. SUMMARY MODEL FOR P2P SYSTEMS

In this section, we first present our summary model architecture and the key assumptions adopted for building data summaries in hybrid P2P systems. Second, we describe the summarization technique that allows generating these data summaries. Then, we formally define the notion of data summary in a P2P network.

2.1 Model Architecture

Data indexes are maintained in P2P systems using one of the following approaches. A *centralized* approach maintains a global index over all the data shared in the network, and thus provides a centralized-search facility [18]. A *hybrid decentralized* approach distributes indexes among some specialized nodes (e.g. supernodes), while a *pure decentralized* approach distributes indexes among all the participants in the network (e.g. structured DHTs, Routing Indices). Each of these approaches provides a different trade-off between the cost of maintaining the indexes and the benefits obtained for queries. In our work, we have adopted the second approach since it is the only one that exploits peer heterogeneity, which is a central key to allow P2P systems scaling up without compromising their decentralized nature. The architecture of our summary model is presented in Figure 1. First of all, we assume that peers store and share structured data, more specifically relational databases. Using a summarization technique, each peer maintains a local summary of its own database. Then, peers that belong to the same domain, i.e. peers that are associated to the same superpeer, cooperate to maintain a global (merged) summary over their shared data. The set of global materialized summaries and links between the corresponding domains, provide a virtual *complete* summary, which ideally describes the content of all data shared in the network. Peers that are willing to cooperate in order to benefit from such a global summary, are supposed to agree on a common, shared data representation. As described later in Section 2.2.1, summaries are represented in a user-defined vocabulary. Thus, the heterogeneity of summary representations make difficult their exploitation (e.g. summary merging operation in Section 3).

As introduced, our work mainly target collaborative database applications that involve a distributed, huge amount of semantically rich data. In such a context the assumption of a common data representation, referred later as a *Common Background Knowledge* (CBK), seems not to be a strength constraint, since the number of participants is supposed to be limited, and thus the agreement on a common description of data semantics is feasible. An example of a CBK in a medical collaboration is the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) [19], which provides a common language that enables a consistent way of capturing, sharing and aggregating health data across specialties and sites of care.

The next section briefly describes the summarization process that generates summaries of relational databases with interesting features, making it scalable in a distributed environment.

2.2 Summarization Process

A summarization process is integrated to each peer's DataBase Management System (DBMS) to allow constructing the local summary level of Figure 1. Our approach is based on SaintEtiQ [12], an online linguistic approach for summarizing databases. The SaintEtiQ system takes tabular data as input and produces multi-resolution summaries of records through a two-step process: online mapping and summarization. For illustration, consider the following relational database which is reduced to a single *Patient* relation (Table 1).

Id	Age	Sex	BMI	Disease
t_1	15	*female*	17	*Anorexia*
t_2	20	*male*	20	*Malaria*
t_3	18	*female*	16.5	*Anorexia*

Table 1: Raw data

2.2.1 Mapping Service

The SaintEtiQ system relies on Zadeh's fuzzy set theory [24] and, more specifically on linguistic variables [25] and fuzzy partitions [26] to represent data in a concise form. The fuzzy set theory is used to translate records according to a *Background Knowledge (BK)* provided by the user. The Background Knowledge BK is a priori built over the attributes that are considered relevant to the summarization process. In the above relation, the selected attributes are age and bmi[1]. Basically, the mapping operation replaces the original values of every record in the table by a set of linguistic descriptors defined in the BK. For instance, with a linguistic variable on the attribute age (Figure 2), a value

[1]Body Mass Index (BMI) attribute: patient's body weight divided by the square of the height.

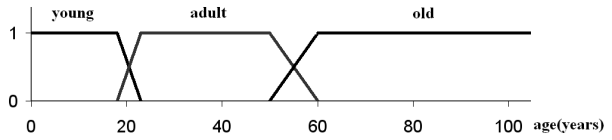

Figure 2: Fuzzy Linguistic Partition on age

$t.AGE = 20$ years is mapped to $\{0.3/adult,\ 0.7/young\}$ where 0.3 is a membership grade that tells how well the label *adult* describes the value 20. Extending this mapping to all the attributes of a relation could be seen as locating the overlapping cells in a grid-based multidimensional space that map records of the original table. The fuzzy grid is provided by BK and corresponds to the user's perception of the domain.

Thus, tuples of Table 1 are mapped into three distinct grid-cells denoted by c_1, c_2 and c_3 in Table 2. A priori, the fuzzy label *underweight* provided by the BK on attribute BMI, perfectly matches (with degree 1) range $[15, 17.5]$, while the fuzzy label *normal* perfectly matches range $[19.5, 24]$ of raw values. Besides, tuple count column gives the proportion of records that belongs to the cell and $0.3/adult$ says that *adult* fits the data only with a small degree (0.3). It is computed as the maximum of membership grades of tuple values to *adult* in c_3.

Id	Age	BMI	tuple count
c_1	*young*	*underweight*	2
c_2	$0.7/young$	*normal*	0.7
c_3	$0.3/adult$	*normal*	0.3

Table 2: Grid-cells mapping

The fuzziness in the vocabulary definition of *BK* permits to express any single value with more than one fuzzy descriptor and thus avoid threshold effect thanks to the smooth transition between different categories. Besides, BK leads to the point where tuples become indistinguishable and then are grouped into grid-cells such that there are finally many more records than cells. Every new (coarser) tuple stores a record count and attribute-dependent measures (min, max, mean, standard deviation, etc.). It is then called a *summary*.

2.2.2 Summarization Service

The summarization service is the last and the most sophisticated step of the SaintEtiQ system. It takes *grid-cells* as input and outputs a collection of summaries hierarchically arranged from the most generalized one (the root) to the most specialized ones (the leaves) [12]. Summaries are clusters of grid-cells, defining hyperrectangles in the multidimensional space. In the basic process, leaves are grid-cells themselves and the clustering task is performed on K cells rather than N tuples ($K << N$).

From the mapping step, cells are introduced continuously in the hierarchy with a top-down approach inspired of D.H. Fisher's Cobweb [22], a conceptual clustering algorithm. Then, they are incorporated into best fitting nodes descending the tree. Three more operators could be apply, depending on partition's score, that are *create*, *merge* and *split* nodes. They allow developing the tree and updating its current state. Figure 3 represents the summary hierarchy built from the cells c_1, c_2 and c_3.

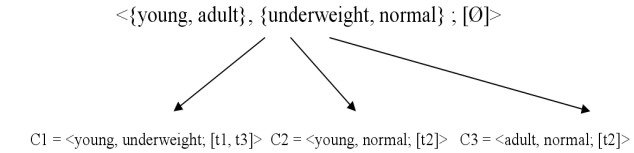

Figure 3: Example of SaintEtiQ hierarchy

2.2.3 Scalability Issues

Memory consumption and time complexity are the two main factors that need to be taken care of in order to guaranty the capacity of the summary system to handle massive datasets. First, the time complexity of the SaintEtiQ process is in $O(K)$, where K is the number of cells to incorporate into a hierarchy of summaries. Here we note that, the number of cells that are produced by the mapping service depends only on the granularity and the fuzziness of the BK definition. A fine-grained and overlapping BK will produce much more cells than a coarse and crisp one. Besides, an important feature is that in the summary algorithm, raw data have to be parsed only once, and this are processed with a low time cost. Second, the system requires low memory consumption for performing the summary construction algorithm as well as for storing the produced summaries.

Thanks to these advantages (i.e. linear time complexity and controlled memory consumption), we believe that this system is scalable in a distributed environment, and promises a successful integration in P2P systems. More details on the summary construction/maintenance costs are presented in Section 5.1.1

2.3 Summary Representation

In this section, we introduce basic definitions related to the summarization process.

Definition 1. **Summary** Let $E = \langle A_1, \ldots, A_n \rangle$ be a n-dimensional space equipped with a grid that defines basic n-dimensional areas called cells in E. Let R be a relation defined on the cartesian product of domains D_{A_i} of dimensions A_i in E. Summary z of relation R is the bounding box of the cluster of cells populated by records of R.

The above definition is constructive since it proposes to build generalized summaries (hyperrectangles) from cells that are specialized ones. In fact, it is equivalent to performing an *addition* on cells:

$$z = c_1 + c_2 + \ldots + c_p$$

where $c_i \in L_z$, the set of p cells (summaries) covered by z.

A summary z is then an *intentional description* associated with a set of tuples R_z as its *extent* and a set of cells L_z that are populated by records of R_z.

Thus, summaries are areas of E with hyperrectangle shapes provided by BK. They are nodes of the summary tree built by the SaintEtiQ system.

Definition 2. **Summary Tree** A summary tree is a collection S of summaries connected by $\preccurlyeq$, the following partial order:

$$\forall z, z' \in \mathcal{Z}, \quad z \preccurlyeq z' \iff R_z \subseteq R_{z'}$$

The above link between two summaries provides a generalization/specialization relationship. And assuming that summaries are hyperrectangles in a multidimensional space, the partial ordering defines *nested summaries* from the larger one to the single cells. General trends in the data could be identified in the very first levels of the tree whereas precise information has to be looked at near the leaves.

For our purpose, we also consider a summary tree as an indexing structure over distributed data in a P2P system. Thus, we add a new dimension to the definition of a summary z: a *peer-extent* P_z, which provides the set of peers having data described by z.

Definition 3. **Peer-extent** Let z be a summary in a given hierarchy of summaries S, and P the set of all peers who participated to the construction of S. The *peer-extent* P_z of the summary z is the subset of peers owning, at least, one record of its extent R_z: $P_z = \{p \in P \mid R_z \cap R_p \neq \emptyset\}$, where R_p is the view over the database of node p, used to build summaries.

Due to the above definition, we extend the notion of *data-oriented* summary in a given database, to a *source-oriented* summary in a given P2P network. In other words, our summary can be used as a database index (e.g. referring to relevant tuples), as well as a semantic index in a distributed system (e.g. referring to relevant nodes).

The summary hierarchy S will be characterized by its *Coverage* in the P2P system; that is, the number of data sources described by S. Relative to the hierarchy S, we call *Partner Peer* a peer whose data is described by at least a summary of S.

Definition 4. **Partner peers** The set of *Partner peers* P_S of a summary hierarchy S is the union of peer-extents of all summaries in S: $P_S = \{\cup_{z \in S} P_z\}$.

For simplicity, in the following we designate by "summary" a hierarchy of summaries maintained in a P2P system, unless otherwise specified.

3. SUMMARY MANAGEMENT

In this section, we present our algorithms for summary construction and maintenance in a given domain. First, we work in a static context where all participants remain connected. Then we address the volatility of peers and propose appropriate solutions.

3.1 Summary Construction

We assume that each global summary GS is associated with a *Cooperation List* (CL) that provides information about its partner peers. An element of CL is composed of two fields. A partner identifier *PeerID*, and a *2*-bit freshness value v that provides information about the description freshness as well as the availability of the corresponding database.

- value 0 (initial value): the descriptions are fresh relative to the original data,
- value 1: the descriptions need to be refreshed,
- value 2: the original data are not available. This value is used while addressing peer volatility in Section 3.3.

The construction algorithm starts at a superpeer (referred later as Summary Peer SP) who broadcasts a SUMPEER message with a given value of TTL (e.g. $TTL = 2$). This message contains SP's identifier, and a hop value h initialized to 0, which is used to compute the distances between SP and its clients.

A peer p who received a first SUMPEER message, maintains information about the corresponding summary peer SP. Then, p sends to SP a LOCALSUM message that contains its local summary LS, and thus becomes a partner peer in the SP's domain. Upon receiving this last message, SP merges LS to its current global summary GS, and adds a new element in the cooperation list. However, a peer p who is already a partner may receive a new SUMPEER message. In that case, only if the new summary peer is *closer* than the old one (based on latency), it chooses to drop its old partnership through a DROP message, and it proceeds to participate to the new domain. Here, we note that one could use another metric to compute distances between nodes (e.g. the similarity between summaries). We now suppose that a peer p does not belong to any domain (is not a partner peer), and wants to participate to a global summary construction. Using a *selective walk*, it can rapidly find a summary peer SP. A selective walk is a random walk where a peer chooses intentionally the highest-degree neighbor at each query forwarding step [23]. The information about SP, which is maintained at each of its partners, makes the selective walk even shorter. Once a partner or a summary peer is reached, the FIND message is stopped.

3.2 Summary Maintenance

A critical issue for any indexing structure is to maintain the index, relative to the current data instances, without incurring high costs. For a local summary, it has been demonstrated that the summarization process guarantees an incremental maintenance, using a *push* mode for exchanging data with the DBMS, while performing with a low complexity [29]. In this section, we propose a strategy for maintaining a global summary in a given domain, based on both *push* and *pull* techniques, in order to minimize the number of messages exchanged in the system. The appropriate algorithm is divided into two phases: data modification and summary reconciliation.

3.2.1 Push: Data Modification

Let GS be a global summary and P_{GS} the set of its partner peers. Each peer in P_{GS} is responsible for refreshing its own element in the GS's cooperation list. A partner peer p observes the modification rate issued on its local summary LS. When LS is considered as enough modified, the peer p sets its freshness value v to 1, through a *push message* to the corresponding summary peer SP. The value 1 indicates that the local summary version being merged while constructing GS does not correspond any more to the current instance of the database.

An important feature is that the frequency of push messages depends on modifications issued on local summaries, rather than on the underlying databases. It has been demonstrated in [29] that, after a given process time, a summary hierarchy becomes very stable. As more tuples are processed, the need to adapt the hierarchy decreases and hopefully, once all existing attribute combinations have been processed, incorporating new tuple consists only in sorting it

in a tree. A summary modification can be detected by observing the appearance/disappearance of descriptors in summary intentions.

3.2.2 Pull: Summary Reconciliation

The summary peer SP, in its turn, observes the fraction of old descriptions (i.e. number of ones) in the cooperation list. Whenever this fraction exceeds a threshold value α, the global summary GS must be refreshed. The threshold α is our system parameter, which is tuned to provide the desired trade-off between summary freshness and summary updating cost. More analysis is given in Section 5.1. To update its global summary, SP pulls all the partner peers to merge their current local summaries into a new version of GS, which will be then under reconstruction. The algorithm is described as follows. SP initiates a reconciliation message that contains a new summary *NewGS* (initially empty). The message is propagated from a partner to another (started at SP). When a partner p receives this message, it first merges *NewGS* with its local summary. Then, it sends the message to another partner (chosen from the cooperation list CL). If p is the last visited peer, it sends the message to SP who will store the new version of the global summary. All the freshness values in CL are reset to zero. This strategy distributes the charge of summary merging on all partners, instead of imposing on SP to receive all local summaries and to make the merging calculations alone. Furthermore, this strategy guarantees a high availability of the global summary, since only one update operation is performed at the end by SP.

3.3 Peer Dynamicity

In large P2P systems, a peer connects mainly to download some data and may leave the system without any constraint. Therefore, the shared data can be submitted to a low modification rate, while the rate of node arrival/departure is very important. We now study the effect of this peer dynamicity on our summary management algorithms, and propose appropriate solution.

In unstructured P2P systems, when a new peer p joins the system, it contacts some existing peers to determine the set of its neighbors. If one of these neighbors is a partner peer, p sends its local summary LS to the corresponding summary peer SP, and thus becomes a new partner in the SP's domain. When a partner peer p decides to leave the system, it first sets its freshness value v to two in the cooperation list, through a push message. This value reminds the participation of the disconnected peer p to the corresponding global summary, but also indicates the unavailability of the original data. There are two alternatives to deal with such a freshness value. First, we can keep the data descriptions and use it, when a query is approximately answered using the global summary. A second alternative consists in considering the data descriptions as expired, since the original data are not accessible. Thus, a partner departure will accelerate the summary reconciliation initiating. In the rest of this paper, we adopt the second alternative and consider only a *1*-bit freshness value v: a value 0 to indicate the freshness of data descriptions, and a value 1 to indicate either their expiration or their unavailability. However, if peer p failed, it could not notify its summary peer by its departure. In that case, its data descriptions will remain in the global summary until a new summary reconciliation is executed. The reconciliation algorithm does not require the participation of a disconnected peer. The global summary GS is reconstructed, and descriptions of unavailable data will be then omitted.

Now, when a summary peer SP decides to leave the system, it sends a release message to all its partners using the cooperation list. Upon receiving such a message, a partner p makes a selective walk to find a new summary peer. However, if SP failed, it could not notify its partners. A partner p who has tried to send push or query messages to SP will detect its departure and thus search for a new one.

4. QUERY PROCESSING

In this section, we describe how a query Q, posed at a peer p, is processed. Peer p first sends Q to the summary peer SP of its domain. SP proceeds then to query the available global summary GS. As highlighted in the introduction and all along this paper, summary querying allows to achieve two distinct tasks depending on the user/application requirements: *peer localization* to return the original results, and *approximate answering* to return approximate answers. Summary querying is divided into two phases: 1) query reformulation and 2) query evaluation.

4.1 Query Reformulation

First, a selection query Q must be rewritten into a flexible query Q^* in order to be handled by the summary querying process. For instance, consider the following query Q on the Patient relation in Table 1:

```
select age from Patient where sex = ‘‘female’’
and BMI < 19 and disease = ‘‘anorexia’’
```

This phase replaces the original value of each selection predicate by the corresponding descriptors defined in the Background Knowledge (BK). Therefore, the above query is transformed to Q^*:

```
select age from Patient where sex = ‘‘female’’
BMI in {underweight,normal} and disease = ‘‘anorexia’’
```

Let QS (resp.QS^*) be the *Query Scope* of query Q (resp.Q^*) in the domain, that is, the set of peers that should be visited to answer the query. Obviously, the query extension phase may induce false positives in query results. To illustrate, a patient having a BMI value of 20 could be returned as an answer to the query Q^*, while the selection predicate on the attribute `BMI` of the original query Q is not satisfied. However, false negatives cannot occur, which is expressed by the following inclusion: $QS \subseteq QS^*$.

In the rest of this paper, we suppose that a user query is directly formulated using descriptors defined in the BK (i.e. $Q = Q^*$). As we discussed in the introduction of this work, a doctor that participates to a given medical collaboration, may ask query Q like "the *age* of *female* patients diagnosed with *anorexia* and having an *underweight* or *normal* BMI". Thus, we eliminate potential false positives that may result from query extension.

4.2 Query Evaluation

This phase deals with matching a set of summaries organized in a hierarchy S, against the query Q. The query is transformed into a logical proposition P used to qualify the link between each summary and the query. Proposition P is under a conjunctive form in which all descriptors appears

as literals. In consequence, each set of descriptors yields on corresponding clause. For instance, the above query Q is transformed to P = *(female) AND (underweight OR normal) AND (anorexia)*. A valuation function has been defined to valuate the proposition P in the context of a summary z. Then, a selection algorithm performs a fast exploration of the hierarchy and returns the set Z_Q of most abstract summaries that satisfy the query. For more details see [31]. Once Z_Q determined, the evaluation process can achieve two distinct tasks: 1) Peer localization, and 2) Approximate answering.

4.2.1 Peer Localization

Since the extended definition of a summary z provides a peer-extent, i.e. the set of peers P_z having data described by its intent (see Definition 3), we can define the set P_Q of relevant peers for the query Q as follows: $P_Q = \{\cup_{z \in Z_Q} P_z\}$. The query Q is directly propagated to these relevant peers. However, the efficiency of this query routing depends on the completeness and the freshness of summaries, since stale answers may occur in query results. We define a *False Positive* as the case in which a peer p belongs to P_Q and there is actually no data in the p source that satisfies Q (i.e. $p \notin QS$). A *False Negative* is the reverse case in which a p does not belong to P_Q, whereas there exists at least one tuple in the p data source that satisfies Q (i.e. $p \in QS$).

4.2.2 Approximate Answering

A distinctive feature of our approach is that a query can be processed entirely in the summary domain. An approximate answer can be provided from summary descriptions, without having to access original, distributed database records. The selected summaries Z_Q are aggregated according to their interpretation of proposition P: summaries that have the same required characteristics on all predicates (i.e. *sex*, *BMI* and *disease*) form a class. The aggregation in a given class is a union of descriptors: for each attribute of the selection list (i.e. *age*), the querying process supplies a set of descriptors which characterize summaries that respond to the query through the same interpretation [31]. For example, according to Table 1, the output set obtained for the two classes {*female*, *underweight*, *anorexia*}, and {*female*, *normal*, *anorexia*} is *age* = {*young*}. In other words, *all* female patients diagnosed with *anorexia* and having an *underweight* or *normal* *BMI* are *young* girls.

In the case where exact answers are required, suppose now that processing a query Q in a given domain d_i returns C_i results, while the user requires C_t results. We note that, if C_t is less than the total number of results available in the network, Q is said to be a *partial-lookup* query. Otherwise, it is a *total-lookup* query. Obviously, when C_i is less than C_t, the query should be propagated to other domains. To this end, we adopt the following variation of the flooding mechanism.

Let P_i the subset of peers that have answered the query Q in the domain d_i: $|P_i| = (1 - FP) \cdot |P_Q|$, where FP is the fraction of false positives in query results. The query hit in the domain is given by: $(|P_i| \,/\, |d_i|)$. As shown by many studies, the existing P2P networks have small-world features [2]. In such a context, users tend to work in groups. A group of users, although not always located in geographical proximity, tends to use the same set of resources (i.e. *group locality* property). Thus, we assume that the probability of finding answers to query Q in the neighborhood of a relevant peer in P_i, is very high since results are supposed to be *nearby*. This probability is also high in the neighborhood of the originator peer p since some of its neighbors may be interested in the same data, and thus have cached answers to similar queries. Such assumptions are even more relevant in the context of interest-based clustered networks. Therefore, the summary peer SP_i of domain d_i sends a flooding request to each peer in P_i as well as to peer p. Upon receiving this request, each of those peers sends the query to its neighbors that do not belong to its domain, with a limited value of TTL. Once a new domain is reached or TTL becomes zero, the query is stopped. Besides, the summary peer SP sends the request to the set of summary peers it knows in the system. This will accelerate covering a large number of domains. In each visited domain, the query is processed as described above. When the number of query results becomes sufficient (i.e. larger than C_t), or the network is entirely covered, the query routing is terminated.

5. PERFORMANCE EVALUATION

In this section, we devise a simple model for the summary management cost. Then, we evaluate our model by simulation using the BRITE topology generator and SimJava.

5.1 Cost Model

A critical issue in summary management is to trade off the summary updating cost against the benefits obtained for queries.

5.1.1 Summary Update Cost

Here, our first undertaking is to optimize the update cost while taking into account *query accuracy*. In the next section, we discuss query accuracy which is measured in terms of the percentage of false positives and false negatives in query results. The cost of updating summaries is divided into: usage of peer resources, i.e. time cost and storage cost, and the traffic overhead generated in the network.

Time Cost. A unique feature of SaintEtiQ is that the changes in the database are reflected through an incremental maintenance of the summary hierarchy. The time complexity of the summarization process is in $O(K)$ where K is the number of cells to be incorporated in that hierarchy [29]. For a global summary update, we are concerned with the complexity of merging summaries. The Merging method that has been proposed is based on the SaintEtiQ engine. This method consists in incorporating the leaves L_z of a given summary hierarchy S_1 into an another S_2, using the same algorithm described by the SaintEtiQ summarization service (referenced in Section 2.2.3). It has been proved that the complexity C_{M12} of the Merging(S_1, S_2) process is constant w.r.t the number of tuples [27]. More precisely, C_{M12} depends on the maximum number of leaves of S_1 to incorporate into S_2. However, the number of leaves in a summary hierarchy is not an issue because it can be adjusted by the user according to the desired precision. A detailed Background Knowledge (BK) will lead to a greater precision in summary description, with the natural consequence of a larger summary. Moreover, the hierarchy is constructed in a top-down approach and it is possible to set the

summarization process so that the leaves have any desired precision.

Storage Cost. We denote by k the average size of a summary z. In the average-case assumption, there are $\sum_{i=0}^{d} B^i = (B^{d+1}-1)/(B-1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus the average space requirement is given by: $C_m = k.(B^{d+1}-1)/(B-1)$. Based on real tests, $k = 512$ bytes gives a rough estimation of the space required for each summary. An important issue is that the size of the hierarchy is quite related to its stabilization (i.e. B and d). As more cells are processed, the need to adapt the hierarchy decreases and incorporating a new cell may consist only in sorting a tree. Hence, the structure of the hierarchy remains stable and no additional space is required. On the other hand, when we merge two hierarchies S_1 and S_2 having sizes of C_{m1} and C_{m2} respectively, the size of the resultant hierarchy is always in the order of the $max\,(C_{m1}, C_{m2})$. However, the size of a summary hierarchy is limited to a maximum value which corresponds to a maximum number of leaves that cover all the possible combinations of the BK descriptors. Thus, storing the global summary at the summary peer is not a strength constraint.

According to the above discussion, the usage of peer resources is optimized by the summarization process itself, and the distribution of summary merging while updating a global summary. Thus, we restrict now our focus to the traffic overhead generated in the P2P network.

Network Traffic. Recall that there are two types of exchanged messages: *push* and *reconciliation*. Let local summaries have an average lifetime of L seconds in a given global summary. Once L expired, the node sends a (push) message to update its freshness value v in the cooperation list CL. The reconciliation algorithm is then initiated whenever the following condition is satisfied: $\sum_{v \in CL} v/|CL| \geq \alpha$, where α is a threshold that represents the ratio of old descriptions tolerated in the global summary. During reconciliation, only one message is propagated among all partner peers until the new global summary version is stored at the summary peer SP. Let F_{rec} be the reconciliation frequency. The update cost is:

$$C_{up} = 1/L + F_{rec} \quad messages\ per\ node\ per\ second \qquad (1)$$

In this expression, $1/L$ represents the number of push messages which depends either on the modification rate issued on local summaries or the connection/disconnection rate of peers in the system. Higher is the rate, lower is the lifetime L, and thus a large number of push messages are entailed in the system. F_{rec} represents the number of reconciliation messages which depends on the value of α. This threshold is our system parameter that provides a trade-off between the cost of summary updating and query accuracy. If α is large, the update cost is low since a low frequency of reconciliation is required, but query results may be less accurate due both to false positives stemming from the descriptions of non existent data, and to false negatives due to the loss of relevant data descriptions whereas they are available in the system. If α is small, the update cost is high but there are few query results that refer to data no longer in the system, and nearly all available results are returned by the query.

5.1.2 Query Cost

When a query Q is posed at a peer p, it is first matched against the global summary available at the summary peer SP of its domain, to determine the set of relevant peers P_Q. Then, Q is directly propagated to those peers. The query cost in a domain d is given by:

$$C_d = (1 + |P_Q| + (1 - FP) \cdot |P_Q|) \quad messages,$$

where $(1 - FP) \cdot |P_Q|$ represents the query responses messages (i.e. query hit in the domain).

Here we note that, the cooperation list CL associated with a global summary provides information about the relevance of each database description. Thus, it gives more flexibility in tuning the *recall/precision* trade-off of the query answers in domain d. The set of all partner peers P_H in CL can be divided into two subsets: $P_{old} = \{p \in P_H \mid p.v = 1\}$, the set of peers whose descriptions are considered old, and $P_{fresh} = \{p \in P_H \mid p.v = 0\}$ the set of peers whose descriptions are considered fresh according to their current data instances. Thus, if a query Q is propagated only to the set $V = P_Q \cap P_{fresh}$, then precision is maximum since all visited peers are certainly matching peers (no false positives), but recall depends on the fraction of false negatives in query results that could be returned by the set of excluded peers $P_Q \backslash P_{fresh}$. On the contrary, if the query Q is propagated to the extended set $V = P_Q \cup P_{old}$, the recall value is maximum since all matching peers are visited (no false negatives), but precision depends on the fraction of false positives in query results that are returned by the set of peers P_{old}.

Now we consider that the selectivity of query Q is very high, such that each relevant peer has only one result tuple. Thus, when a user requires C_t tuples, we have to visit C_t relevant peers. The cost of inter-domain query flooding is given by:

$$C_f = ((1 - FP) \cdot |P_Q| + 2) \cdot \sum_{i=1}^{TTL} k^i \quad messages,$$

where k is the average degree value (e.g. average degree of 3.5, similar to Gnutella-type graphs). Remember that, the set of relevant peers who have answered the query (i.e. $(1 - FP) \cdot |P_Q|$), the originator and the summary peers participate to query flooding. In this expression, we consider that a summary peer has on average k long-range links to k summary peers. As a consequence, the total cost of a query is:

$$C_Q = C_d \cdot \frac{C_t}{(1-FP)\cdot|P_Q|} + C_f \cdot (1 - \frac{C_t}{(1-FP)\cdot|P_Q|}) \qquad (2)$$

In this expression, the term $C_t/((1 - FP) \cdot |P_Q|)$ represents the number of domains that should be visited. For example, when $C_t = ((1 - FP) \cdot |P_Q|)$, one domain is sufficient and no query flooding is required.

5.2 Simulation

We evaluated the performance of our solutions through simulation, based on the above cost model. First, we describe the simulation setup. Then we present simulation results to evaluate various performance dimensions and parameters: scale up, query accuracy, effect of the freshness threshold α.

5.2.1 Simulation Setup

We used the SimJava package [10] and the BRITE universal topology generator [14] to simulate a power law P2P network, with an average degree of 4. The simulation parameters are shown in Table 3 and fall into three categories: network parameters, workload parameters, and system parameters. In our tests, we consider that local summary lifetimes are quite related to the node lifetimes, since the rate

Parameter	value
local summary lifetime L	skewed distribution, Mean=3h, Median=1h
number of peers n	16–5000
number of queries q	200
matching nodes/query *hits*	10%
freshness threshold α	0.1–0.8

Table 3: Simulation Parameters

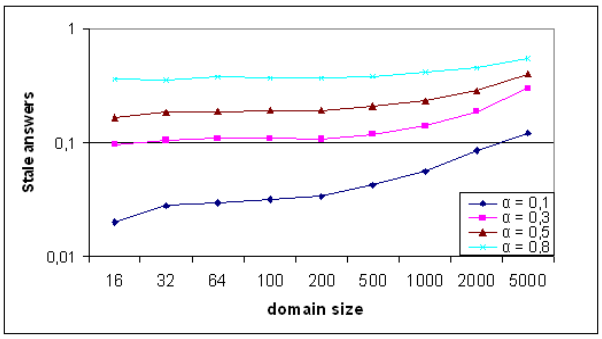

Figure 4: Stale answers vs. domain size

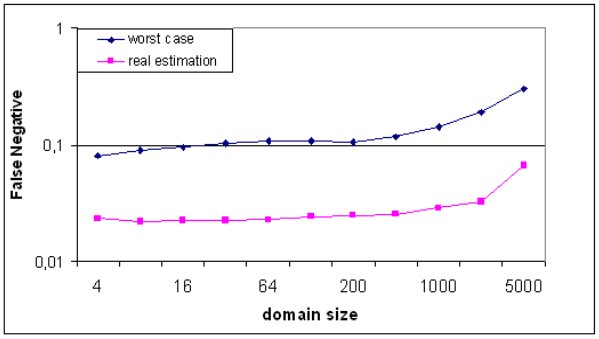

Figure 5: False negative vs. domain size

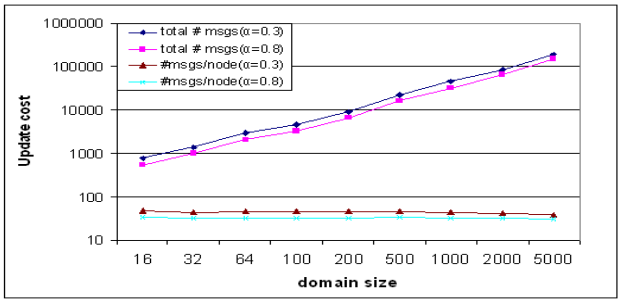

Figure 6: Number of messages vs. domain size

of node connection/disconnection is supposed to be greater than the modification rate issued on local summaries, and this for two reasons. First, in large P2P systems, we mainly deal with selection queries to locate and download required data. Thus, the original data are submitted to a low modification rate. Second, our summaries are even more stable than the original data (as we discussed before). Thus, the volatility of peers is the main reason for a global summary reconciliation. Under this assumption, we consider that local summary lifetimes, like node lifetimes, follow a skewed distribution with a mean value of 3 hours, and a median value of 60 minutes. Our workload has 200 queries. The query rate is 0.00083 queries per node per second (1 query per node per 20 mns) as suggested in [5]. Each query is matched by 10% of the total number of peers. Finally, Our system parameter α that decides of the reconciliation frequency varies between 0.1 and 0.8.

5.2.2 *Update Cost*

In this set of experiments, we quantify the trade-off between query accuracy and the cost of updating a global summary in a given domain. Figure 4 depicts the fraction of stale answers in query results for different values of the threshold α. Here, we illustrate the worst case. For each partner peer p having a freshness value equal to 1, if it is selected in the set P_Q then it is considered as false positive. Otherwise, it is considered as false negative. However, this is not the real case. Though it has a freshness value equal to 1, the peer p does not incur stale answers unless its database is changed relative to the posed query Q. Thus, Figure 4 shows the worst, but very reasonable values. For instance, the fraction of stale answers is limited to 11% for a domain of 500 peers when the threshold α is set to 0.3 (30% of the peers are tolerated to have old/non existent descriptions).

As mentioned in Section 5.1.2, if we choose to propagate the query only to the set $V = P_Q \cap P_{fresh}$ we eliminate the possible false positives in query results. However, this may lead to additional false negatives. Figure 5 shows the fraction of false negatives in function of the domain size. Here we take into account the probability of the database modification relative to the query, for a peer having a freshness value equal to 1. We see that the fraction of false negatives is limited to 3% for a domain size less than 2000. The real estimation of stale answers shows a reduction by a factor of 4.5 with respect to the preceded values.

Figure 6 depicts the update cost in function of the domain size, and this for two threshold values. The total number of messages increases with the domain size, but not surprisingly, the number of messages per node remains almost the same. In the update cost equation 5.1.1, the number of push messages for a given peer is independent of domain size.Besides, the number of reconciliation messages decreases slowly with the number of peers, for a given value of the threshold α. More interestingly, when the threshold value decreases (from 0.8 to 0.3) we notice a little cost increasing of 1.2 on average. However, a small value of the threshold α allows to reduce significantly the fraction of stale answers in query results, as seen in Figure 4. We conclude therefore that tuning our system parameter, i.e. the threshold α, do not incur additional traffic overhead, while improving query accuracy.

5.2.3 *Query Cost*

In this set of experiments, we compare our algorithm for query processing against centralized-index and pure non-index/flooding algorithms. A centralized-index approach is very efficient since a single message allows locating relevant data. However, a central index is vulnerable to attack and it is difficult to keep it up-to-date. Flooding algorithms are

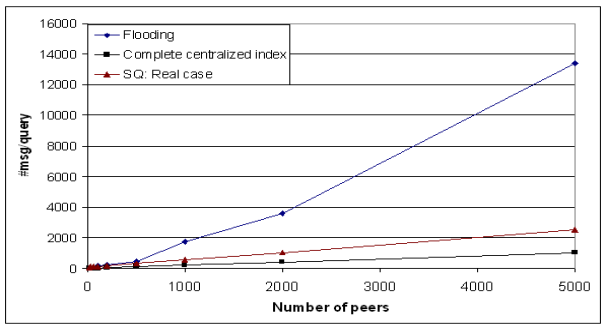

Figure 7: Query cost vs. number of peers

very used in real life, due to their simplicity and the lack of complex state information at each peer. A pure flooding algorithm consists in broadcasting the query in the network till a stop condition is satisfied, which may lead to a very high query execution cost. Here, we limit the flooding by a value 3 of TTL (Time-To-Live).

According to Table 3, the query hit is 10% of the total number of peers. For our query processing approach, which is mainly based on summary querying (SQ), we consider that each visited domain provides 10% of the number of relevant peers (i.e. 1% of the network size). In other words, we should visit 10 domains for each query Q. From equation 5.1.2, we obtain: $C_Q = (10 \cdot C_d + 9 \cdot C_f)$ messages. Figure 7 depicts the number of exchanged messages to process a query Q, in function of the total number of peers. The centralized-index algorithm shows the best results that can be expected from any query processing algorithm, when the index is complete and consistent, i.e. the index covers the totality of data available in the system, and there are no stale answers in query results. In that case, the query cost is: $C_Q = 1+2\cdot((0.1)\cdot n)$ messages, which includes the query message sent to the index, the query messages sent to the relevant peers and the query response messages returned to the originator peer p.

In Figure 7, we observe that our algorithm SQ shows good results by significantly reducing the number of exchanged messages, in comparison with a pure query flooding algorithm. For instance, the query cost is reduced by a factor of 3.5 for a network of 2000 peers, and this reduction becomes more important with a larger-sized network. We note that in our tests, we have considered the worst case of our algorithm, in which the fraction of stale answers of Figure 4 occurs in query results (for $\alpha = 0.3$).

6. RELATED WORK

Much research effort has focused on improving the search efficiency in P2P networks through designing good routing and lookup protocols. Gnutella-like systems mainly provide file-name search facility, i.e. a user has to know the file's unique name. Furthermore, the search is done in a *blind* fashion (e.g. Random Walk, modified BFS), trying to propagate a user query to a sufficient number of nodes in order to satisfy it. To enhance search efficiency, some works have designed search algorithms operating on hybrid P2P architectures. Examples of search protocols are GUESS [6] that builds upon the notion of ultrapeers, and Gnutella2 [15]. The third generation of P2P systems employs structured topologies and uses Distributed Hash Table (DHT) functionalities to provide a tight coupling between hosting nodes and data indexes (e.g. [30], [20]). Although these systems offer an efficient search, they compromise peer autonomy since they mandate a specific network structure, and only support point queries. Data is located using unique and globally known data identifiers; complex queries are difficult to support. Current works on P2P systems aim to employ *content-based* routing strategies, since the content of data can be exploited to more precisely guide query propagation. These strategies require gathering information about the content of peer's data. However, the limits on network bandwidth and peer storage, as well as the increasing amount of shared data, call for effective summarization techniques. These techniques allow exchanging compact information on peer's content, rather than exchanging original data in the network.

Existing P2P systems have used keyword-based approaches to summarize text documents. For instance, in [1] documents are summarized by keyword vectors, and each node knows an approximate number of documents matching a given keyword that can be retrieved through each outgoing link (i.e. Routing Indices RIs). Although the search is very bandwidth-efficient, RIs require flooding in order to be created and updated, so the method is not suitable for highly dynamic networks. Other works (e.g. [9]) investigate Vector Space Model (VSM) and build *Inverted Indexes* for every keyword to cluster content. In this model, documents and queries are both represented by a vector space corresponding to a list of orthogonal term vectors called *Term Vector*. The drawback of VSM is its high cost of vector representations in case of P2P churns. In [13], a *semantic-based* content search consists in combining VSM to Latent Semantic Index (LSI) model [7] to find semantically relevant documents in a P2P network. This work is based on hierarchical summary structure over hybrid P2P architecture, which is closely related to what we are presenting in this paper. However, instead of representing documents by vector models, we describe structured data (i.e. relational database) by synthetic summaries that respect the original data schema (i.e. synthetic tuples representation).

To the best of our knowledge, none of the summarization techniques used in P2P systems allows for an *approximate query answering*. All works have focused on facilitating content-based query routing, in order to improve search efficiency. In this work, we propose a summarization technique that produces synthetic, multidimensional views over structured data. We believe that the novelty of our approach relies on the fact that our data summaries allow for a semantic-based query routing, as well as for an approximate query answering as we mentioned in the introduction. Our approach for database summarization is based on SaintEtiQ [29], which relies on the fuzzy set theory to build robust summaries.

7. CONCLUSION

In this paper, we proposed a model for summary management in unstructured P2P systems. The innovation of this proposal consists in combining the P2P and database summarization paradigms, in order to support data sharing on a world wide scale. The database summarization approach that we proposed provides efficient techniques for data localization as well as for data description in P2P systems.

In fact, our summaries are semantic indexes that support locating relevant data based on their content. Besides, an important feature is that these summaries are compact data descriptions that can approximately answer a query without retrieving original records from huge, highly distributed databases. We made the following contributions. First, we defined an appropriate summary model for hybrid P2P systems, where a domain is defined as being the set of a super-peer and its clients. Then, we proposed efficient algorithms for summary management in a given domain. Our performance evaluation showed that the cost of query routing in the context of summaries is significantly reduced in comparison with flooding algorithms, without incurring high costs of summary maintenance.

8. REFERENCES

[1] A.Crespo and H.G.Molina. Routing indices for peer-to-peer systems. In *Proc. of the 28 tn Conference on Distributed Computing Systems*, July 2002.

[2] A.Iamnitchi, M.Ripeanu, and I.Foster. Locating data in (small-world?) peer-to-peer scientific collaborations. In *IPTPS*, pages 232–241, 2002.

[3] A.Oser, F.Naumann, W.Siberski, W.Nejdl, and U.Thaden. Semantic overlay clusters within super-peer networks. In *Proc of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing in Conjunction with the VLDB*, 2003.

[4] A.Rowstron and P.Druschel. Storage management and caching in PAST, a large–scale, persistent peer-to-peer storage utility. In *Proc.SOSP*, 2001.

[5] B.Yang and H.G.Molina. Comparing hybrid peer-to-peer systems. In *Proc VLDB*, 2001.

[6] B.Yang, P.Vinograd, and H.Garcia-Molina. Evaluating guess and non-forwarding peer-to-peer search. In *ICDCS '04*, 2004.

[7] C.Papadimitriou, H.Tamaki, P.Raghavan, and S.Vempala. Latent semantic indexing: A probabilistic analysis. In *ACM Conference on Principles of Database Systems (PODS)*.

[8] C.Wang, L.Xiao, Y.Liu, and P.Zheng. Dicas: An efficient distributed caching mechanism for p2p systems. *IEEE Transactions on Parallel and Distributed Systems*, 2006.

[9] F.Cuenca-Acuna, C.Peery, R.Martin, and T.Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *HPDC-12*, 2003.

[10] F.Howell and R.McNab. Simjava: a discrete event simulation package for java with the applications in computer systems modeling. In *Int. Conf on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation*, 1998.

[11] G.Koloniari, Y.Petrakis, and E.Pitoura. Content–based overlay networks of xml peers based on multi-level bloom filters. In *Proc VLDB*, 2003.

[12] G.Raschia and N.Mouaddib. A fuzzy set-based approach to database summarization. *Fuzzy sets and systems 129(2)*, pages 137–162, 2002.

[13] H.Shen, Y.Shu, and B.Yu. Efficient semantic-based content search in p2p network. *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 2004.

[14] http://www.cs.bu.edu/brite/.

[15] http://www.gnutella2.com.

[16] http://www.gnutella.com.

[17] http://www.kazaa.com.

[18] http://www.napster.com.

[19] http://www.snomed.org/snomedct.

[20] I.Stoica, R.Morris, D.Karger, M.F.Kaashoek, and H.Balakrishnan. Chord: A scalabale peer-to-peer lookup service for internet applications. In *Proc ACM SIGCOMM*, 2001.

[21] I.Tartinov and *et al*. The Piazza peer data management project. In *SIGMOD*, 2003.

[22] K.Thompson and P.Langley. Concept formation in structured domains. In *Concept formation: Knowledge and experience in unsupervised learning*, pages 127–161. Morgan Kaufmann.

[23] L.Adamic and *et* al. Search in power law networks. *Physical Review E*, 64:46135–46143, 2001.

[24] L.A.Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[25] L.A.Zadeh. Concept of a linguistic variable and its application to approximate reasoning-I. *Information Systems*, 8:199–249, 1975.

[26] L.A.Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100:9–34, 1999.

[27] M.Bechchi, G.Raschia, and N.Mouaddib. Merging distributed database summaries. In *ACM Sixteenth Conference on Information and Knowledge Management (CIKM)*, 2007.

[28] R.Akbarinia, V.Martins, E.Pacitti, and P.Valduriez. Design and implementation of appa. In *Global Data Management (Eds. R. Baldoni, G. Cortese and F. Davide)*. IOS press, 2006.

[29] R.Saint-Paul, G.Raschia, and N.Mouaddib. General purpose database summarization. In *Proc VLDB*, pages 733–744, 2005.

[30] S.Ratnasamy, P.Francis, M.Handley, R.M.Karp, and S.Shenker. A scalable content–addressable network. In *SIGCOMM*, 2001.

[31] W.A.Voglozin, G.Raschia, L.Ughetto, and N.Mouaddib. Querying the SAINTETIQ summaries–a first attempt. In *Int.Conf.On Flexible Query Answering Systems (FQAS)*, 2004.