# 19. Schema-Based Peer-to-Peer Systems

Wolfgang Nejdl, Wolf Siberski (L3S and University of Hannover)

## 19.1 Introduction

When sharing information or resources — the most prominent application of Peer-to-Peer systems — one is immediately faced with the issue of searching. Any application which provides an information collection needs some means to enable users finding relevant information. Therefore, the expressivity of the query language supported by the system is a crucial aspect of Peer-to-Peer networks. Daswani et al. [154] distinguish key-based, keyword-based and schema-based systems.

*Key-based* systems can retrieve information objects based on a unique hash-key assigned to it. This means that documents for example have to be requested based on their name. This kind of queries is supported by all DHT networks (cf. Chapter 7). Typically, key-based search features are not exposed to end-users, but rather used as basic infrastructure.

*Keyword-based* systems extend this to the possibility to look for documents based on a list of query terms. This means that users do not have to know the document they are looking for, but can ask for all documents relevant to particular keywords. Non-ranking keyword-based systems find matching resources by executing a string or string pattern matching algorithm, e.g. on the file name. Ranking keyword-based approaches score documents according to their relevance depending on statistics derived from document full text. Chapter 20 describes the latter kind of systems.

*Schema-based* systems manage and provide query capabilities for structured information. *Structured* means that the information instances adhere to a predefined schema. For example, in a digital archive any document is described using a schema consisting of elements as title, author, subject, etc. In schema-based systems, queries have to be formulated in terms of the schema (e.g. "find all documents with author=Smith"). Nowadays the dominant schema-based systems are relational databases; other important variants are XML and Semantic Web data stores. Schema-based Peer-to-Peer systems are sometimes also called *Peer Data Management Systems* (e.g., in [273]).

Digital archives are an application area where schema-based queries provide significant value. Here, users often need to formulate complex queries, i.e., queries with constraints regarding several criteria, to specify their search. For example, to find recent books about Java programming, one would need to exclude outdated books and to disambiguate between the Java programming language ("find all books where 'Java' occurs in the title, publication

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 323-336, 2005.
© Springer-Verlag Berlin Heidelberg 2005

date is less than three years ago, and subject is a subtopic of computer"). Such complex queries are only supported by schema-based systems.

We can observe two converging development lines, one regarding database systems, the other in Peer-to-Peer networks. Databases started as centralized systems, where one server processes queries from all clients. Since then, they have evolved towards a higher degree of distribution, e.g by introducing mediator-based [622] distributed query processing[1]. At the same time Peer-to-Peer systems have developed towards support for more expressive queries [445, 70, 4, 274, 308]. Schema-based Peer-to-Peer systems are the point where these two directions of research meet, as shown in figure 19.1(see also [260]).

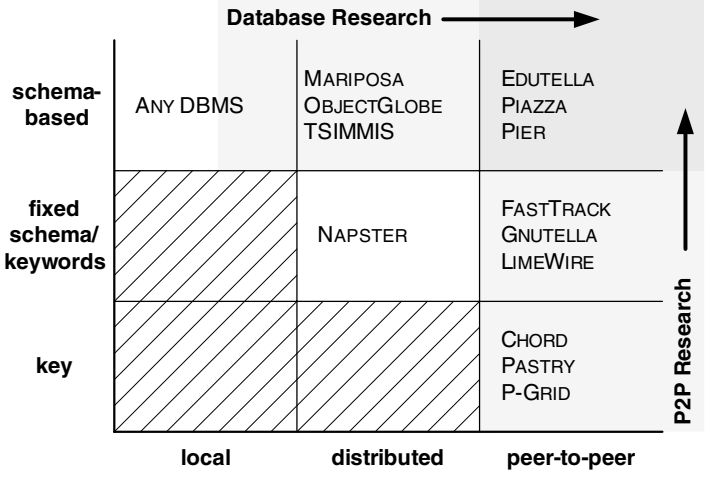


**Fig. 19.1:** Schema Capabilities and Degree of Distribution

Schema-based Peer-to-Peer systems inherit characteristics of database systems as well as Peer-to-Peer networks:

- *Strict adherence of data to a schema* All data stored at the peers is structured according to a schema. Note that not necessarily all peers share the same schema. For example, a peer storing information about courses, professors and students will use another schema than a digital archive peer storing information about scientific documents. This leads to the requirement of schema mapping in Peer-to-Peer systems (see 19.4)
- *Schema-based query expressions* Query constraints as well as requested information are specified with reference to schema elements. In a heterogeneous network, i.e., where more than one schema is used, queries will be sent to a subset of peers only, depending on the query constraints.

[1] A good overview of distributed database technology is [363].

– *No global knowledge* Maintaining global information such as a schema repository doesn't scale if the network grows, because this information would have to be replicated to all peers, and the update frequency grows with network size. Therefore query processing needs to rely on local information only, and we can't use techniques from distributed databases which require a central mediator.
– *Self-organization and -administration* A Peer-to-Peer network has no administrator who could create indexes, materialized views, etc., as in a central database. All management activities necessary to improve query processing need to be executed in a self-organizing fashion (cf. Chapter 16).

The chapter is structured as follows: Several design dimensions for schema-based Peer-to-Peer systems can be distinguished. These dimensions and possible choices for each dimension are presented in section 19.2. Semantic Web standards allow to represent information in a structured way which is especially suited for sharing. Therefore, they are a perfect basis for a schema-based Peer-to-Peer network. 19.3 exemplifies the connections between design choices by describing an existing Peer-to-Peer system for the Semantic Web. 19.4 includes the advanced topics schema mapping, distributed query plans and top-$k$ query processing.

## 19.2 Design Dimensions of Schema-Based Peer-to-Peer Systems

Several decisions influence the design of a schema-based Peer-to-Peer system: the choice of the data model and associated query language, how to distribute the data among the peers (data placement), how to connect the peers (topology), and the query routing algorithm all determine search capabilities and performance [154]. This section describes options for each such design dimension along with their consequences for the resulting system.

### 19.2.1 Data Model and Query Language

The Data model used to store information is tightly connected to the aspect of the query language. Many data models have been proposed for storing structured data and it is out of scope to discuss them in detail. We rather want to mention some basic distinctions with respect to the data model that influence the ability of the system. The most basic way of storing structured data is in terms of a fixed, standardized schema that is used across the whole system. In this view, less complex data models like the one used in key- or keyword-based systems can be considered as special case of a very simple fixed schema. Despite the obvious limitations, fixed schema approaches are often observed in Peer-to-Peer systems because this eliminates the problem of

schema interoperability. Interoperability is a problem in systems that allow the user to define and use a local schema. This does not only ask for a suitable integration method, it also leads to maintenance problems, because local schemas can evolve and new schemas can be added to the system when new peers join.

Nevertheless, several schema-based Peer-to-Peer systems which support flexible and/or heterogeneous schemas have been developed in the last years. Nearly all of them have employed common and widespread data models to benefit from well-defined semantics and a lot of experience. The most prevalent are the *relational* (e.g. PIER [308]), the *XML* (e.g. Piazza [273]) and the *Semantic Web* (e.g. Edutella [445], REMINDIN [586]) data model. In these models schemas can be viewed a collection of type definitions which in turn consist of attribute definitions. Taking the relational model as example, table definitions specify types and column definitions the attributes associated with each type.

Usually, one or several default query languages are associated with a data model. For the relational model this is SQL, for XML it is XPath and XQuery (see 19.3.1 for a discussion of query languages for the Semantic Web). To improve query processing efficiency, in schema-based Peer-to-Peer systems often only a subset of the associated query language is supported.

### 19.2.2 Data Placement

The data placement dimension is about where the data is stored in the network. Two different strategies for data placement in the network can be identified: placement according to ownership and placement according to search strategy.

*Placement according to ownership.* In a Peer-to-Peer system it seems most natural to store information at the peer which is controlled by the information owner. And this is indeed the typical case. The advantage is that access and modification are under complete control of the owner. For example, if the owner wants to cease publishing of its resources, he can simply disconnect his peer from the network. In the owner-based placement approach the network is only used to increase access to the information.

*Placement according to search strategy.* The complementary model is the when peers do not only cooperate in the search process, but already in storing the information. Then the network as a whole is like a uniform facility to store and retrieve information. In this case, data is distributed over the peers so that it can be searched for in the most efficient manner, i.e. according to the search strategy implemented in the network. Thus, the owner has less control, but the network becomes more efficient.

Both variants can be further improved in terms of efficiency by the introduction of additional caching and replication strategies. Note that while

this improves the network performance, it also reduces the owner's control of information.

### 19.2.3 Topology and Routing

As with all Peer-to-Peer systems, the choice of topology and corresponding routing algorithms is crucial for schema-based networks. The distinction between unstructured and structured topologies is significant here, too. Hierarchical approaches such as super-peer networks can be of specific use for schema-based networks, and therefore are described separately.

*Structured Networks.* As shown in Chapter 7, structured networks, especially DHTs allow very efficient access for known keys. Thus, they are a good foundation for the creation of indexes. However, one DHT overlay can't serve for indexing more than one schema element (e.g. one column of a table). To answer queries containing arbitrary constraints one would need to maintain a separate DHT for every attribute. PIER [308] is such a system. It allows for efficient querying, but only for queries based on indexed attributes. For each of them, PIER maintains a DHT overlay network. This works good if network and/or data changes are limited, but at the price of increased maintenance effort otherwise.

*Unstructured Networks.* In unstructured networks, evaluation of complex queries is much simpler. They are forwarded to relevant peers and fully processed there. Each peer sends back its results, and all result sets are finally merged at the originating peer. A representative such networks is the Piazza system [273].The obvious drawback of such an approach is that some kind of flooding algorithm has to be used to distribute the query within the network. Traditional limitation techniques like time-to-live do only work well if the data is significantly replicated. This is common for file-sharing networks, but does not necessarily apply for information-sharing systems. A better way to reduce query distribution is to let peers apply filters on their connections for each query and send it only in the direction of relevant peers. The relevancy can be determined either based on a content summary provided by each peer [446] or based on the results of previous query evaluations [586].

Introducing short-cuts increases the probability further that a query reaches all relevant peers. Here, all peers continually asses their current connections based on the results of previous queries. If connections didn't yield enough results, they are given up. As replacement, the peer establishes new direct (short-cut) connections to peers relevant for past queries. Over time, this leads to an optimized topology [586]. Interestingly, specific reconnection strategies can lead to the emergence of regular topologies, although not enforced by the network algorithms [542]. This is characteristic for self-organizing systems in other areas (like biology) too, and seems to be a promis-

ing middle way between pure structured and pure unstructured networks (cf. Chapter 15).

*Super-Peer Networks.* Inspired by the mediator work in distributed databases, a special kind of hybrid networks, so-called super-peer networks have gained attention as topology for schema-based Peer-to-Peer networks. The distribution of peer performance characteristics (processing power, bandwidth, availability, etc.) is not distributed uniformly over all peers in a network. Exploiting these different capabilities in a Peer-to-Peer network can lead to an efficient network architecture [635], where a small subset of peers, called super-peers, takes over specific responsibilities for peer aggregation, query routing and possibly mediation. For this purpose, only the super-peers form a Peer-to-Peer network, and all other peers connect directly to the resulting super-peer backbone.

Super-peer-based Peer-to-Peer infrastructures usually exploit a two-phase routing architecture, which routes queries first in the super-peer backbone, and then distributes them to the peers connected to the super-peers. Like database mediators, super-peers only need to know which schema elements each connected peer supports. This is a small amount of information and thus easily indexed and maintained. Another advantage is the ability of super-peers to perform coordinating tasks as creating a distributed query plan for a query (see 19.4.2). The disadvantage of super-peer networks is the need to dedicate explicitly specific nodes to the super-peer role which limits the self-organization capabilities of the network.

When ontologies are used to categorize information, this can be exploited to further optimize peer selection in a super-peer network. Each super-peer becomes responsible for one or several ontology classes. Peers are clustered at these super-peers according to the classes of information they provide. Thus, an efficient structured network approach can be used to forward a query to the right super-peer, which distributes it to all relevant peers [395].

*Discussion.* Structured and unstructured networks have complementary advantages and disadvantages regarding their use for schema-based networks. The predetermined structure allows for more efficient query distribution in a structured network, because each peer 'knows' the network structure and can forward queries just in the right direction. But this does only work well if query complexity is limited, otherwise too many separate overlay networks have to be created and maintained.

In unstructured networks, peers do not know exactly in which direction to send a query. Therefore, queries have to be spread within the network to increase the probability of hitting the peer(s) having the requested resource, thus decreasing network efficiency. On the other hand, queries can take more or less any form, as long as each peer is able to match its resources against them locally. For support of highly expressive queries, as needed e.g. in ontology-based systems, only unstructured networks are feasible. An exception are some DHT systems which have been extended recently into

hybrid networks that also support flooding-based query distribution strategies [106, 393].

Super-peer networks can alleviate the performance issues of pure unstructured topologies, but at the price of introducing two distinct peer classes.

## 19.3 Case Study: A Peer-to-Peer Network for the Semantic Web

To get an insight of all the issues which have to be solved in a schema-based Peer-to-Peer system it is instructive to take a thorough look at an existing system and the design choices involved in building it. As shown below, the Semantic Web data model is especially suited for sharing of structured data. Therefore a system from that context – Edutella – has been selected as case study.

The aim of the Edutella project [186, 445] is to design and implement a schema-based Peer-to-Peer infrastructure for the Semantic Web. Edutella relies on W3C Semantic Web standards [360, 91] to describe distributed resources, and uses basic Peer-to-Peer primitives provided as part of the JXTA framework ([255], see also 21.3.1).

### 19.3.1 Semantic Web Data Model and Query Language

In the Semantic Web, an important aspect for its overall design is the exchange of data among computer systems without the need of explicit consumer-producer relationships. The Resource Description Format standard (RDF, [360]) is used to annotate resources on the Web and provide the means by which computer systems can exchange and comprehend data. All resources are identifiable by unique resource identifiers (URIs plus anchor ids). All annotations are represented as statements of the form <subject, property, value>, where subject identifies the resource we want to describe (using a URI), property denotes which attribute we specify, and value the attribute value, expressed as a primitive datatype or an URI referring to another resource. For example, to annotate document `http://site/sample.html` with its author, we could use the statement <`http://site/sample.html` *dc:creator* ``Paul Smith''>.

RDF Schema (RDFS, [91]) is used to define the vocabulary used for describing our resources. RDFS schema definitions include resource classes, properties and property constraints (domain, range, etc.). For example, property *dc:creator* is a property of the standardized Dublin Core metadata schema for document archives [159]. We can use any properties defined in the schemas we use, possibly mix different schemas, and relate different re-

sources to each other, when we want to express interdependencies between these resources, hierarchical relationships, or others.

Another important characteristic of RDF metadata is the ability to use distributed annotations for one and the same resource. In contrast to traditional (non-distributed) database systems, it is not necessary to store all annotations of a resource on one server. One server might collect detailed information about authors, e.g., affiliation and contact information. Other servers which hold document metadata could just reference author URIs defined as part of the author servers data and don't need to repeat that information locally. This ability for distributed allocation of metadata makes RDF very suitable for the construction of distributed repositories.

In the Semantic Web context two types of query languages have been developed, SQL-like and rule-based languages. The former has the advantage that users familiar with SQL very quickly become familiar with the query language. Essentially, they allow the formulation of constraints on the RDF data graph and thus extraction of matching subgraphs.

But, although RDF data are basically graphs, query languages based on simple graph matching and subgraph extraction are not sufficient: they cannot reason about the semantics underlying such data, given in the form of schema languages like RDFS or OWL [407]. Even if we have a query language that takes RDFS into account, this built-in support for exactly one fixed schema language is not sufficient, as it does not allow us to query and combine RDF data expressed in multiple schema languages which is necessary in the case of distributed scenarios where providers can neither be forced to use the same schema nor the same schema language. Therefore, more expressive query formalisms have been investigated, which usually build on rule-like languages [78, 445, 564].

It is therefore required that a query language supports the definition of the semantics of several schema languages. This can appropriately be done with rule languages based on Datalog (or Horn logic in general). In Edutella, the Query Exchange Language (QEL, [451]) provides us with an expressive query exchange language which serves as a common query interchange format, into which local query languages can be translated (quite a common approach in distributed databases). Edutella peers are connected to the network using a wrapper-based architecture, where the wrapper is responsible for translating local query languages into the Edutella common query model.

Edutella peers can be highly heterogeneous in terms of the functionality they offer. In order to handle different query capabilities, Edutella defines several QEL language compliance levels, describing which kind of queries a peer can handle (conjunctive queries, relational algebra, transitive closure, etc.) Obviously, restricting query expressiveness allows for better optimization of query processing, so e.g. for publish/subscribe systems based on such a Peer-to-Peer network, a quite restricted form of queries is sufficient [116]. However, all peers still can use the same internal query representation for

all capability levels, thus enabling reuse of existing functionality, e.g., for translation purposes.

### 19.3.2 Schema-Based Routing Indices

Edutella uses a super-peer topology, where super-peers form the network backbone and take care of routing queries through the network [446]. Only a small percentage of nodes are super-peers, but these are assumed to be highly available nodes with high computing capacity. In the Edutella network they are arranged in the HyperCuP topology [541].

The HyperCuP algorithm is capable of organizing peers into a recursive graph structure from the family of Cayley graphs, out of which the hypercube is the most well-known topology. This topology allows for $\log_2 N$ path length and $\log_2 N$ number of neighbors, where $N$ is the total number of nodes in the network (i.e. the number of super-peers in our case). The algorithm works as follows: All edges are tagged with their dimension in the hypercube. A node invoking a request sends the message to all its neighbors, tagging it with the edge label on which the message was sent. Nodes receiving the message forward it only via edges tagged with higher edge labels (see [541] for details).

The Edutella super-peers employ routing indices, which explicitly acknowledge the semantic heterogeneity of schema-based Peer-to-Peer networks, and therefore include schema information as well as other possible index information. This super-peer backbone is responsible for message routing and integration of metadata. Super-peers in the Edutella network are arranged in the HyperCuP topology discussed in the last section.

Peers connect to the super-peers in a star-like fashion, providing content as well as content metadata. Figure 19.2 shows a very simple example of such a backbone. Alternatives to this topology are possible, as long as they guarantee the spanning tree property for the super-peer backbone, which is required for maintaining our routing indices and distributed query plans [94]. Other topologies are possible for other kinds of indices [137].

*Super-Peer/Peer Routing Indices.* Edutella super-peers characterize their associated peers using super-peer/peer routing indices. Whenever a new peer connects to a super-peer, it sends a self-description, including some meta-information about available data, during the initial handshake. The super-peer uses this self description to create indices at different granularities which are later on used to select appropriate peers for answering incoming queries. The following indices are always maintained:

– *Schema Index.* A base assumption is that different peers will support different schemas. These schemas are uniquely identified by their respective namespace, therefore the SP/P routing index contains the schema identifier and the peers supporting the respective schema.

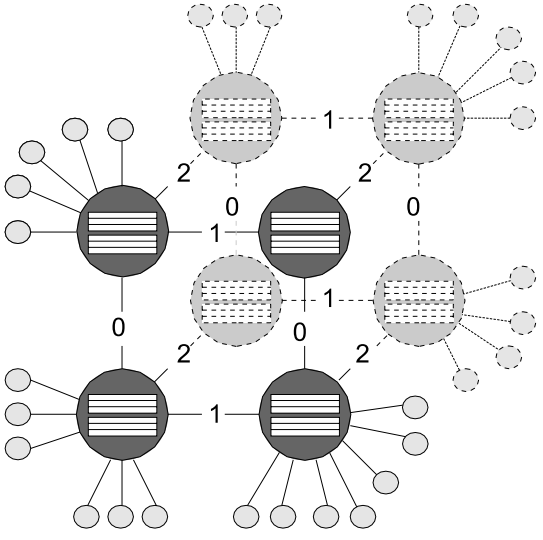


**Fig. 19.2:** HyperCuP Super-Peer Topology

– *Property/Sets of Properties Index.* This Routing index contains properties or sets thereof thus enabling peers to support only parts of schemas. If a query only refers to the supported property subset, it is forwarded to the corresponding peer.

Additionally super-peers can use even more fine-grained indexes, e.g. a *Property Value Range Index* or even a *Property Value Index* to restrict the set of relevant peers even further, at the price of increased index size.

These SP/P indices are updated when a peer connects to a super-peer, and contain all necessary information about connected peers. Entries are valid only for a certain time, and are deleted when the peer does not renew/update it regularly (e.g., because it leaves the network). Peers notify the super-peer when their content changes in ways that trigger an update of the index.

*Super-Peer/Super-Peer Routing Indices.* As with peers, queries should not be sent to all super-peers. To achieve this goal super-peer/super-peer routing indices can be used to route among the super-peers. These SP/SP indices are essentially extracts and summaries from the local SP/P indices. They contain the same kind of information as SP/P indices, but refer to the (direct) neighbors of a super-peer. Queries are forwarded to super-peer neighbors based on the SP/SP indices (restricting the basic HyperCuP broadcast), and sent to connected peers based on the SP/P indices.

Update of Edutella SP/SP indices is based on the registration (or update) messages from connected peers. Whenever an SP/P index changes, this change is propagated to (potentially) all super-peers using a (reversed)

HyperCuP broadcast. Whenever an SP/SP index stays the same after the update, propagation stops.

Because one important aspect of Peer-to-Peer networks is their dynamicity, the SP/SP indices are not, in contrast to distributed architectures in the database area (e.g., [88]), replicated versions of a central index, but rather parts of a distributed index similar to routing indices in TCP/IP networks.

When a query arrives at a super-peer, it matches the schema elements occurring in the query against the index information. The query is only forwarded to the peers and along the super-peer connections which use the same schema elements and are therefore able to deliver results. Thus, the indices act as message forwarding filter which ensure that the query is distributed only to relevant peers.

## 19.4 Advanced Topics

### 19.4.1 Schema Mapping

Mappings that explicitly specify the semantic relation between information objects in different sources are the basis for the integration of information from different sources. Normally, such mappings are not defined between individual data objects but rather between elements of the schema. Consequently, the nature of the mapping definitions strongly depend on the choice of a schema language. The richer the schema language, the more possibilities exist to clarify the relation between elements in the sources. However, both creation and use of mappings becomes more complex with the increasing expressiveness. There are a number of general properties mappings can have that influence their potential use for information integration:

- Mappings can relate single objects from the different information sources or connect multiple elements that are connected by operators to form complex expressions.
- Mappings can be undirected or directed and only state the relation from the point of view of one of the sources connected.
- Mappings can declaratively describe the relation between elements from different sources or consist of a procedural description of how to convert information from one source into the format of the other
- Declarative mappings can be exact or contain some judgement of how correct the mapping reflects the real relation between the information sources

In the context of Peer-to-Peer information sharing, the use of mappings is currently restricted to rather simple mappings. Most existing systems use simple equality or subsumption statements between schema elements. Approaches that use more complex mappings (in particular conjunctive queries) do not scale to a large number of sources. A prominent example is the Piazza approach [273].

### 19.4.2 Distributed Query Plans

Currently, data processing in Peer-to-Peer networks works as follows: The Peer-to-Peer network is queried for information satisfying some given conditions, and this query is routed to the peers which can answer it. When the possibly large number of results from distinct data sources is returned to the client peer, further query processing takes place centrally at the client. On the other side, data integration systems such as ObjectGlobe [88] distribute query plans to the distributed hosts as much as possible and thus are able to place operators close to the data sources. To generate the query plans, however, these systems need to know where all data are located.

The naive and straightforward way to combine both approaches is to first use Peer-to-Peer capabilities to find out where data is stored, and then use this information to generate a distributed query plan at the client. This query plan becomes instantiated and executed on different hosts. But this would be not very efficient. Based on the super-peer architecture described in section 19.3.2 the query execution can be optimized by pushing *abstract query plans* through the super-peer-based network, where each super-peer picks and expands those parts of the query plan that can be executed locally [94]. The decision which operations can be executed locally is guided by SP/SP and SP/P indices. This leads to a dynamic *on the fly* distribution and expansion of query plans. Operators are placed next to data sources and thus utilize distributed computing resources more effectively.

The expansion of these abstract query plans can be based on different strategies, related to the quality of clustering in the Peer-to-Peer network. If the data are clustered well with respect to the queries, it is most efficient to push joins in the query as near as possible to the data sources, and then take the union of the results for these joins. If the clustering does not reflect the partitions needed by the query, it is more beneficial to gather the data and do the joins on these data on a central super-peer (see also [349]).

### 19.4.3 Top-$k$ Query Processing

Meaningful querying for information, whether on the Web or in information systems and databases, often retrieves answers together with an indication of how well the results match the query. Various kinds of metadata available through the Semantic Web offer additional semantic information which may be integrated into the retrieval process. However, this generally comes at the price of large result set sizes that are often unmanageable for the individual user, especially because they are in arbitrary order (at least with respect to the relevance for the user). Since users are usually only interested in a few *most relevant* answers, the goal is to return manageable result sets of answers ranked by their relevance according to the query.

Ranking scores each resource that matches a query using a certain set of criteria and then returns it as part of a ranked list. Additionally, we need to restrict the number of results, to make it easier for the user to use the results, and to minimize traffic in a Peer-to-Peer environment. In databases, this approach is referred to as top-$k$ query processing, where only the $k$ best matching resources are returned to the user.

Top-k ranking in Peer-to-Peer networks has to address two additional challenges [444]:

*Mismatch in scoring techniques and input data.* Scoring techniques and input data used by the different peers can have a strong impact on getting the correct overall top-scored objects. Since we want to minimize network traffic, but nevertheless integrate the top-scored objects from all different peers (and super-peers) within each super-peer, each super-peer has to decide how to score answers to a given query. In general we want to assume that every peer throughout the network uses the same methods to score documents with respect to a query, though input data to compute these scores may be different.

*Using only distributed knowledge.* Distributed information and thus different input data to score answers complicates top-k retrieval, because many scoring measures that take global characteristics into account simply cannot be evaluated correctly with limited local knowledge. See section 20.1.2 for a context where some global knowledge (or estimation) is required for correct score calculation.

One algorithm for top-$k$ query evaluation is presented in [54]. It uses the same super-peer architecture as described in section 19.3.2. The algorithm is based on local rankings at each peer, which are aggregated during routing of answers for a given query at the super-peers involved in the query answering process. Each peer computes local rankings for a given query, and returns just the best matches to its super-peer. At the super-peer, the results are merged, using the result scores, and routed back to the query originator. On this way back, each involved super-peer again merges results from local peers and from neighboring super-peers and forwards only the best results, until the aggregated top $k$ results reach the peer that issued the corresponding query. While results are routed through the super-peers, they note in an index the list of peers / super-peers which contributed to the top $k$ results for a query. This information is subsequently used to directly route queries that were answered before only to those peers able to provide top answers. Thus, the distribution of queries can be limited significantly and query processing becomes much more efficient. To cope with network churn, index entries expire after some time, and the query is again sent to all relevant peers. Algorithm details can be found in [54], together with optimality proofs and simulation results.

## 19.5 Conclusion

Schema-based Peer-to-Peer networks are the next step in the evolution of distributed database query processing. We can see two research directions in this area, infrastructures which view the Peer-to-Peer network as one virtual database, and infrastructures for creating a Peer-to-Peer network of independent, possibly heterogeneous data sources.

To create an efficient distributed, but homogeneous database, structured networks are a suitable foundation, although the issue of multi-dimensional queries has not been solved completely. The paradigmatic example for such a system is PIER. Query processing in such systems is very efficient, at the price of restricted query expressivity. A typical characteristic is that data is not placed on the data owner's peer, but distributed according to the underlying network structure.

On the other hand, for largely heterogeneous networks no pure structured network is suitable as underlying topology, because no global schema exists on which a uniform index could be built. Thus, unstructured networks are the prevalent foundation for such networks. Starting from random graphs, algorithms have emerged which optimize peer connections and thus improve query processing performance. However, these systems do not yet scale to more than thousands of participants. Hybrid topologies (e.g. super-peer networks like Edutella) can improve efficiency further by dedicating peers to the traditional database mediator role.

Schema mapping is still a challenging issue in distributed databases, and therefore it is no wonder that highly scalable solutions do not (yet) exist for the Peer-to-Peer case. However, systems like Piazza have already gone far towards loosely-coupled integration of heterogeneous systems in a Peer-to-Peer network.

The need for more information sharing and information integration on the Web and in other open contexts is growing, and schema-based Peer-to-Peer systems are a promising way to meet these needs. While not yet escaped from research labs, we will probably see practical applications based on such infrastructures soon.