A Schema-based Peer-to-Peer Infrastructure for Digital Library Networks

Der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades

Doktor der Naturwissenschaften

Dr. rer. nat.

genehmigte Dissertation von

Dipl.-Inform. Wolf Siberski

geboren am 10. Februar 1966 in Göttingen

2006

Referent:	Prof. Dr. Wolfgang Nejdl
Ko-Referenten:	Prof. Dr. Karl Aberer
	Prof. Dr. Udo Lipeck
Tag der Promotion:	15. Dezember 2006

Ben Zoma said: Who is wise? He who learns from every man, as it is said:
"From all my teachers have I gained understanding"

Pirkei Avot 4,1

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Prof. Dr. Wolgang Nejdl. He introduced me to methodical research, always had time for scientific discussions, gave me the freedom to pursue my research goals, and provided an excellent research environment, to mention just a few points. In short, this thesis would not have been possible without his ample support and guidance.

I also would like to thank my other referees Prof. Dr. Karl Aberer and Prof. Dr. Uwe Lipeck for their very helpful comments and suggestions.

I'm grateful to Prof. Dr. Heinz Züllighoven and Prof. Dr. Christiane Floyd, who have shaped my understanding not only of software, but also of computer science in general.

The collaboration and discussion with my colleagues at L3S Research Center, University of Hannover, and elsewhere was an indispensable source of information and has spawned a lot of insights for this thesis. But what is even more important, our joint work was always a pleasure, and we had a lot of fun together. I would like to thank all of my colleagues for their cooperation and openness, especially Dr. Uwe Thaden, Dr. Wolf-Tilo Balke, and Dr. Peter Dolog.

It is tremendously helpful to work in a smooth administrative and technical environment. Katia Capelli, Thomas Lösch, Dr. Christoph Strutz, Iris Ziesenieß, Claudia Saalbach, and Marko Brosowski provide such an environment for L3S, and were always very supportive when I came to them with my minor or major requests.

During the creation of a thesis, it is probably inevitable to face some stumbling blocks. The guide of Dr. Alexandra Fischer-Flebbe helped me in overcoming mine.

I will always be grateful for the love and care of my parents. They gave me self-confidence and intellectual curiosity, the basis for all my work.

Finally, my wife Susanne and my children Dana and Jona bore it with exceeding patience that I couldn't spend enough time with them, and sustained me every day with their their love and affection.

ABSTRACT

A Schema-based Peer-to-Peer Infrastructure for Digital Libraries

in
English

In today's connected world, users are not content with searching only one local library or archive, but want and need to take a substantial number of collections into account when looking for relevant information. Currently, most digital libraries and catalog systems only support local search, and only few facilities offer federated search over several libraries. One reason is that central federation instances cause significant infrastructure costs, and there are only limited incentives for libraries to offer such services. An appealing solution is to avoid a central federation instance and use a completely distributed infrastructure instead, thus also distributing the infrastructure efforts. In this thesis, we will present such an infrastructure which combines peer-to-peer, distributed database and Semantic Web technology to provide seamless search in an open network of digital libraries.

The proposed solution is based on a super-peer topology, where the most powerful nodes form a network backbone and take over mediator-like responsibilities to distribute queries and merge results. The network content is modeled as a database fragmented over all nodes. Our basic algorithm, SPQR (super-peer-based query routing), allows processing of queries according to the classic relational algebra, and is shown to always produce the correct result set with respect to this fragmented database. We present an implementation of our approach which enables the interconnection of library systems conforming to established Open Archive Initiative standards. An extension of SPQR for preference-based queries allows users to retrieve 'best matches' for their queries instead of only exact matches. Extensive evaluations based on a peer-to-peer simulation framework show the algorithm's performance and scalability.

Keywords: peer-to-peer networks, distributed databases, digital libraries

ABSTRACT

A Schema-based Peer-to-Peer Infrastructure for Digital Libraries

in

Deutsch

Die heutige Vernetzung bringt es mit sich, dass Nutzer von Bibliotheken und Archiven sich nicht mehr mit einer einzigen Informationsquelle begnügen, wenn sie nach relevanter Information suche, sondern eine mehr oder weniger große Anzahl von Informationsanbietern konsultieren wollen und müssen. Momentan unterstützen die meisten Katalogsysteme und digitalen Bibliotheken nur lokale Suche, und es gibt nur eine geringe Anzahl von Serviceangeboten für föderierte Suche über viele Bibliotheken hinweg. Ein Grund dafür ist, dass solche Services merkliche Infrastrukturkosten mit sich bringen, und es für jede einzelne Bibliothek wenig Anreize gibt, diese Kosten zu tragen. Eine attraktive Lösung für diese Problematik ist, zentrale Services ganz zu vermeiden, und stattdessen eine vollständig verteilte Infrastruktur zu verwenden; auf diese Weise werden auch die Aufwendungen für die Infrastruktur über alle beteiligten Bibliotheken verteilt. In dieser Arbeit stellen wir eine solche Infrastruktur vor, die Ansätze aus Peer-to-Peer-Netzwerken, verteilten Datenbanken und dem Semantic Web kombiniert, um transparente Suche in einem offenen Netzwerk digitaler Bibliotheken zu ermöglichen.
Die vorgeschlagene Lösung basiert auf einer Super-Peer-Topologie, in der die leistungsfähigsten Knoten ein Netzwerk-Backbone formen und Mediator-Aufgaben der Verteilung von Anfragen und Zusammenführung der Ergebnisse übernehmen. Die im Netzwerk angebotenen Informationen werden als über alle Knoten fragmentierte Datenbank modelliert. Zur Verarbeitung relationaler Anfragen in dieser verteilten Datenbank dient der Algorithmus SPQR (Super-peer-based Query Routing), dessen Korrektheit gezeigt wird. Weiterhin wird die Implementierung eines auf SPQR basierenden Netzwerks beschrieben, mit dem Bibliothekssysteme vernetzt werden können, die konform zu etablierten Standards der Open Archive Initiative sind. Aufbauend auf SPQR stellen wir einen Algorithmus für die Verarbeitung präferenzbasierter Anfragen vor, der es erlaubt, 'beste Treffer' für Benutzeranfragen zu identifizieren. Umfangreiche Evaluierungen mit Hilfe eines Simulationsframeworks für Peer-to-Peer-Netzwerke zeigen die Effizienz und Skalierbarkeit der präsentierten Algorithmen.

Stichworte: Peer-to-Peer-Netzwerke, Verteilte Datenbanken, digitale Bibliotheken

Contents

Chapter 1

Introduction

Most experts agree that the future of libraries lies in entering the digital information realm. By combining their experience in selection and provisioning of high-quality information with the dissemination opportunities offered by new technology, libraries will be able to play an important role in the information age. Some even go so far to state that *"digital libraries can become the universal knowledge repositories and communication conduits of the future, a common vehicle by which everyone will access, discuss, evaluate, and enhance information of all forms"* [76]. While this may be a bit exaggerating, we can safely assume that it will at least become partly true. A major requirement for this vision is that digital libraries become open not only from a social, but also from a technical point of view [174]. As long as users searching for information have to consult each library separately, it too tedious for them to find relevant documents, and they will favor the classic Web as information source instead. On the other hand, if users could search transparently on a network of interconnected library sites as easy as they can now on the Web, they presumably would often prefer the controlled and thus more reliable content offered by libraries and other managed digital archives.

In this chapter, we identify characteristics of libraries, especially their catalogs, from library history and derive requirements for the envisioned search capabilities. Based on this requirements analysis, we conclude with the problem statement for our work, and an overview of the remaining chapters.

1.1 A Short History of Library Catalogs

Since over 2000 years libraries and archives are formed by their main purpose: preserving documents for later use, and thus also preserving the knowledge embedded in them. For this aim, they need first to ensure safe storage of these documents, and second to provide means

for retrieving documents when requested. We only consider the second point in this thesis. Already the first famous library, the Alexandrina, in existence about from 300 BC to 400 AD, exhibited a structure which is still (of course now in a very developed form) prevalent[1]:

- A unique identification for each document was established. At the Alexandrina, the author's name and a kind of title were used to identify a work[2].
- A catalog was maintained to decouple search for documents from the physical document storage. With catalogs, people can look for relevant items first, and fetch them later (or let librarians fetch them) using their identification. At the Alexandrina, documents were classified into subjects, such as Drama, Laws, Philosophy, History, Medicine, Mathematics, etc. Probably subcategories were also in use, but for them no firm historical evidence exists. The catalog consisted of a document list for each category, containing the author/semi-title pair used as identification.

Since then, both identification and classification have evolved significantly. Regarding identification, with the advent of printed books, the now common bibliographic metadata properties *publisher* and *publication date* complemented the already established *author* and *title* attributes. Over time, more properties were added to this list, but the mentioned core properties are still prevailing.

Until the 19th century, each library had its own rule set for identification. Typical examples where rules widely varied were the issues of books with anonymous author or books published by an institution, but without explicit author or editor. One of the driving forces behind standardization of these rules was the aim to union complete library catalogs. In the 1850s, a team of librarians under Charles Coffin Jewett started to compile a union catalog of US libraries, and found many difficulties in identifying duplicates, due to differing identification policies. Consequently, he formulated a new set of unified rules which formed the starting point for further initiatives reconciling cataloging rules. This was (and is) a very tedious and long-lasting effort. For example, the institution-as-author issue mentioned above was resolved in the US only in 1967 with AACR (Anglo-American Cataloging Rules, [8]). While some minor differences between identification approaches on international level still exist, nowadays an agreement on all substantial issues has been achieved.

The main reason these standardization efforts are so difficult to perform is that no hierarchical organization is imposed on libraries. Each library is funded locally, and between most libraries no strong organizational ties exist. Of course, umbrella organization have been founded, but still libraries have a tendency to cultivate their independence and autonomy.

[1] The description of catalog history presented here mainly follows [185].

[2] At that time, titles were not yet common. Therefore, typically the first few words of the document served as title supplement.

```
LC Control Number:  97002959
Type of Material:   Book (Print, Microform, Electronic, etc.)
Personal Name:      Kant, Immanuel, 1724-1804.
Main Title:         Critique of pure reason / translated and edited
                      by Paul Guyer, Allen W. Wood.
Uniform Title:      Kritik der reinen Vernunft. English
Published/Created:  Cambridge ; New York : Cambridge University Press, 1998.
Related Names:      Guyer, Paul, 1948-
                    Wood, Allen W.
Description:        xi, 785 p. ; 23 cm.
ISBN:               0521354021 (hardcover)
Notes:              Includes bibliographical references (p. 77-80) and index.
Subjects:           Knowledge, Theory of.
                    Causation.
                    Reason.
Series:             Kant, Immanuel, 1724-1804. Works. English. 1992.
Variant Series:     The Cambridge edition of the works of Immanuel Kant
LC Classification:  B2778.E5 G89 1998
Dewey Class No.:    121 21
Language Code:      eng ger
```

Figure 1.1: Sample Library of Congress Catalog Entry

Let's now turn to the classification issue: To keep collections manageable, structuring by subject was the dominating principle from the start. First, each library invented its own classification scheme to bring structure into its collection. Until the enlightening period starting in the 18th century, subject classifications were not very detailed. Monastic libraries often only divided their collection only into religious and secular works. But with the encyclopedia movement in Europe, the vision of ordering all existing knowledge into one coherent system developed, and this had a deep impact on libraries. In Paris, Jacques-Charles Brunet wrote the *Manual du libraire*, originally targeted to book sellers. But the classification scheme described in this book soon became a guide for classification activities of librarians, too. A breakthrough in classification systems was the Dewey Decimal Classification (DDC), published first by Melvil Dewey in 1876. This system forms the core from which several classification schemes still in use evolved. In 1895 Henry LaFontain and Paul Otlet started work on a European version of DDC which resulted in the Universal Decimal Classification (UDC). At about the same time, the Library of Congress librarians adapted the DDC for their purposes, and published a first version of the Library of Congress Classification (LCC) in 1911. The latter classification became predominant in the US, because libraries could buy catalog cards from the Library of Congress containing all bibliographic and classification information for each book published in the US. This freed especially small and medium-sized libraries from the tedious work of classifying new items on their own. A sample Library of Congress catalog entry is shown in Figure 1.1.

In Germany, the effort to create a normative classification started very late (in the 1970s), but nowadays RSWK (Regeln für den Schlagwortkatalog, [152]) and SWD (Schlagwort-Norm-Datei, [176]) are used by most major libraries. Similar initiatives and classifications exist for other countries as well. Of course, differences in culture lead to different design choices in

these classifications. For example, the DDC is much more coarse-grained in German history, German law, and German literature than SWD. It seems unlikely that a unified world-wide subject classification for libraries will ever come into existence.

Libraries were one of the first institutional users of computers. Already in mid-1960s, libraries started to employ the new technology for cataloging purposes [142]. Unfortunately, the development of catalog systems and related metadata schemas took different directions in the US and in Europe. In US, MARC (Machine Readable Cataloging) was developed and established in 1968 [115]; the current version is MARC 21 [112]. In Europe, the Netherlands became leaders of IT usage in libraries, and a group of libraries developed the PICA (Project for Integrated Catalogue Automation) catalog system from 1969 on. This system became also widely used by libraries in Belgium, France, Germany, and later in the UK. Neither the MARC nor the PICA metadata schema is defined by a standard committee; both are proprietary format. However, their widespread use make them de-facto standard schemas for library catalog systems.

As long as each library maintained its electronic catalog independently, the diversity with respect to bibliographic record structure did not pose any obstacle. But with the advent of computer and subsequently library networks, catalog interoperability became crucial. As a remedy, the Dublin Core Metadata Initiative (DCMI) was founded in 1995, with the goal to create a unifying document archive metadata standard. In 1998, the first DCMI standard was released, the Dublin Core Metadata Element Set, Version 1.0 [184]. The current version is [47]. It focuses on the 15 most important metadata attributes, therefore MARC as well as PICA entries can be transformed to DC in a rather straightforward way. DC became the dominant exchange format for document metadata in the internet world. On the other hand, DC only provides a rather limited set of metadata, compared to MARC and PICA. Therefore, it is not going to replace these schemas. With the advent of multimedia documents, libraries face new challenges regarding catalog metadata, because the current catalog standards are text document centric. One solution is to complement existing schemas with new multimedia specific attributes, e.g., from the MPEG-7 standard [113].

Metadata exchange does not only require a unified format, but also a communication protocol between the participating libraries. In 1999, the Open Archive Initiative (OAI, [183]) was founded, with the goal to define such a protocol. OAI released a protocol for metadata harvesting (OAI-PMH, [91]) in 2001. This protocol allows to exchange complete catalogs between archives. To keep these replicas up-to-date in the face of changes, each entry is associated with a timestamp, and on request only items changed after the last update are sent. However, OAI-PMH is only meant to facilitate replication, search in foreign catalogs is not supported. OAI-PMH and other replication protocols (such as Z39.50 [191]) have enabled the creation of large joint catalogs by replication; the most prominent is WorldCat [40]. In the last years,

several small-scale systems for distributed on-line search have been created, e.g. [128, 89, 41], but this hasn't lead to a search protocol standard (yet).

1.2 Digital Libraries

With the advent of the Web, libraries have started to provide not just document metadata in digital form, but also document content. In the late 1980s and early 1990s, several universities established on-line access to articles of selected journals (e.g. Mercury [117]). In 1991, the publisher Elsevier started the project TULIP (The University Licensing Project), which finally provided nine leading universities with articles from over 40 journals [21]. These first initiatives lead to a rapidly growing interest in on-line provision of library collections. First, such collections were called *electronic libraries* [95], but soon the term *digital libraries* was established [94]. Since then, a vast variety of digital libraries has come into existence, mainly providing scientific information, but also offering items from cultural heritage, politics, learning resources, etc. The size of these libraries ranges from small local archives to large article collections (such as the ACM Digital Library with over 50.000 items).

It is hard to specify exactly what constitutes a digital library. Deegan states that *"there are many different kinds of digital libraries creating, delivering and preserving digital objects that derive from many different formats of underlying data, and it is very difficult to formulate a definition that encapsulates all of these"* [49]. Nevertheless, we can at least identify some characteristics (although not all of them have to be fulfilled for every digital library). Of course, it is essential that resources are provided in digital form. In almost all definitions, these resources have to be available via a distributed network (although some view also a document collection on CD or other media as digital library [187]).

The most important difference between digital libraries and any other digital information resource is that their collections are institutionally maintained and organized, with a specific user group in mind. According to Witten, *"it is in the selection and organization of information by librarians that digital libraries are distinguished from the Web"* [187]. However, this institution does not necessarily have to be a library in the traditional sense. Archives, museums, and any other institutions selecting, preserving and providing documents can turn into digital libraries when they start to use the digital realm for information provision.

It is not required that a digital library offers all of its resources in digital form. Often, for a significant fraction of the collection only metadata is provided on-line. For a while, such libraries have been called *hybrid libraries*, but this distinction is not made anymore.

Borgmann points out that every digital library has a technical and a social foundation: *"Digital*

libraries are a set of electronic resources and associated technical capabilities for creating, searching, and using information. In this sense they are an extension and enhancement of information storage and retrieval systems that manipulate digital data in any medium (text, images, sound [...]) and exist in distributed networks.
Digital libraries are constructed – collected and organized – [...] In this sense, they are an extension, enhancement, and integration of a variety of information institutions as physical places where resources are selected, collected, organized, preserved, and accessed in support of a user community." [23][3]

The difference between a traditional and a digital library is increasingly blurring:

- Digital and non-digital documents are often maintained in the same digital catalog
- Libraries provide extended metadata and relations between documents (citations, abstracts, recommendations, etc.) for printed items in the same way as for digital resources.
- For some collections, users can order digitized versions of printed material on-line, and the requested documents are scanned on demand (e.g., subito [173]).

Summarizing, nearly every large library is turning gradually into a digital library. Levy and Marshal [97] even state that the term 'digital' library is unnecessary. They argue that libraries have always adapted to new types of documents, from clay tablets, papyrus rolls, and parchments up to printed books, gramophone records, and tapes. In the same way, they adapt today to the preservation and provision of digital material. This assessment probably will become true in the next few decenniums, when the dust has settled, and methods and technologies for digital libraries have become commonplace.

One of the key opportunities of digital libraries is their integration in a distributed network. This does not only allow online access by users, but also opens up the option of interconnecting the library systems themselves. Therefore, it is not surprising that very early digital library research projects started to tackle the issue of seamless search over a library network, e.g. [90, 149]. In his article about the Berkeley Digital Library Project Wilenski writes: *"For digital libraries to succeed, we must abandon the traditional notion of 'library' altogether. The reason is as follows: The digital 'library' will be a collection of distributed information services; producers of material will make it available, and consumers will find it and use it."*[186].

OAIster [67] is an exemplary recent representative of such digital library integration projects.

[3] Computer scientists in the digital library field have already gained a reputation in neglecting this social factor, and focusing too much on the technological aspects, and we plead guilty in continuing this tradition in the work presented here, with the lame excuse that solutions for technological problems can be solved quite independently from sociological considerations.

OAIster is a provider of technical reports and articles, and integrates documents from over 350 different sources. It employs the OAI Protocol for Metadata Harvesting (cf. 5.1) to collect document metadata from all connected providers, and also fetches the full documents, if possible. The advantage of this approach is that search is possible independently of the availability of connected nodes. However, the disadvantage is that all the work (replicating metadata and content, storage, query processing, etc.) has to be done by one central server. OAIster shares this characteristic with other existing services, such as CiteSeer [60] or Google Scholar [63]. Essentially, we can say that these information providers mimic the way Web search engines work: all information is replicated to huge central services which evaluate queries on these replicas. One can envision such a scenario for libraries as well, where the existing services evolve to huge central archives containing a joined catalog and providing the search entry point to library content, such as current search engines do for the Web.

But we see several disadvantages to the central approach:

- Even Google, the dominant search engine, only indexes a fraction of the Web (not including library catalog information at all), due to the 'Hidden-Web' issue: a lot of Web sites (including practically all library sites) do not provide their contents as static, interlinked pages, but create pages dynamically from their content database, taking into account the current session context (e.g., the user query). These dynamic pages contribute significantly to the Web and often exhibit above-average quality, but they are not captured by Web search engines.

- No one in the library field would like to give the power of such a central instance to any single library provider. The risk of becoming dependent on the good will of such a library search engine would just be to high. Besides, libraries are traditionally organized in a decentralized fashion, where each library has a lot of freedom to adapt its organization to the special needs of its user group. A central instance imposing guidelines on contributing members would be conflicting with this culture.

- From a technical point of view, a top-down-approach integrating all libraries is much more prone to failure than a bottom-up-approach where some libraries (presumable with similar focus) start to form networks, and the interconnection can proceed as far as culture and technology allows.

- From an economic point of view, a centralized infrastructure such as as a Web search engine incurs significant maintenance costs, which no single library can afford. In this context, a solution allowing to distribute infrastructure costs is much more appropriate[4].

Therefore, our vision is a completely decentralized library network [7], based on peer-to-peer

[4] Another option would be a fee-based service, but this contradicts with the 'free access' policy of most libraries

technology. Each library can join this network freely and provide the amount of information it is willing and able to offer. Users can search for relevant documents, without the need to identify relevant libraries themselves; this is done by the self-organizing network. Peer-to-peer networks are especially suited to contexts where budget is limited and no one can (or wants to) take the burden of funding a central coordinating instance [150].

Summary Since their inception, a core activity of libraries is to maintain and provide catalogs of their holdings. To identify documents within these catalogs, they are marked-up with a well established set of metadata information. The Dublin Core metadata standard is established as common denominator to all major metadata schemas for digital libraries. However, it only covers core information and is complemented by other not so common schemas for provision of extended metadata. Regarding subject classification, no common standard exists, and it is difficult to imagine how such a unifying classification could be developed. On the other hand, the predominant Western classification schemes have a lot in common, because all of them stem from the same ancestor, the DDC.

The library field is characterized by a large degree of autonomy. Each library maintains its own catalog, decides on its own about its accessibility guidelines, etc. On the other hand, it has become clear to the major players in the library area, that to compete with the Web as information provider, interconnecting libraries is crucial. Modern catalogs are already available in electronic form, and thus can be made accessible via search techniques, e.g. from the database or information retrieval area. Indeed, most libraries already provide public search interfaces for end-users on the Web. Additionally, not only the catalog data, but more and more the documents themselves become available in electronic form; this increases search and provision options further.

Current initiatives to integrate distributed catalogs and digital document archives are using a central instance. This instance typically replicates all information from connected sources to provide seamless search over all collections. Our vision is to replace such a central coordinating instance with a network of loosely coupled nodes, providing a search service in a cooperative fashion, based on peer-to-peer technology.

1.3 Problem Statement and Outline

Based on digital library characteristics, we can identify the following requirements for the envisioned network:

- Due to the autonomy of libraries we need a completely decentralized network infras-

tructure which allows information providers to join and leave easily.

- This network needs to route and process queries in an efficient manner. In particular, a query should be routed only to relevant providers to avoid needless query processing at library servers.
- Query capabilities offered by this network should be as good as with current online catalog search interfaces.
- To cater to the trend of digital document provision, keyword search on document full text should be supported – preferably in combination with metadata-based search –, returning only the most relevant documents to the user.
- While standardization of metadata schemas has progressed quite well, still different schemas and differences in their usage exist between libraries. Optimally, this heterogeneity is bridged by automatic mapping between queries and each respective library schema.
- The main competitive advantage of libraries is their information quality and control. Therefore, mechanisms to maintain quality and filter out low-quality or spam providers are required.

A complete coverage of these requirements would exceed by far the scope of a thesis. Therefore we focused our research on metadata search and provision which is at the core of (digital) libraries. Our main contribution is an approach for efficient metadata search in a decentralized library network. We designed and implemented a schema-based peer-to-peer network (SPN) which combines techniques from distributed databases and peer-to-peer systems to achieve efficient database-like query processing on metadata in a fully distributed context. Additionally, we present an extension of this approach which supports the notion of 'best matches' and provides combined keyword and metadata search capabilities. The issues of schema mapping and quality maintenance have not been tackled. However, the proposed network allows for coexistence of different catalog schemas.

Outline Chapter 2 gives an introduction to the fields which form the foundations of our work. As mentioned, the most important areas are distributed (relational) databases and peer-to-peer networks. The Semantic Web standards we use for metadata representation are also introduced here. Chapter 3 analyzes the design dimensions of SPNs and outlines possible approaches for each design challenge. The resulting design space is used to review and classify related work. Our own approach to these challenges, super-peer-based query routing (SPQR), is described in Chapter 4. We present the algorithms for indexing and routing and show the results of a simulation-based evaluation. Chapter 5 describes the design and implementation of OAI-

P2P, an infrastructure to interconnect OAI-conformant library systems. The next step toward a full solution is the extension to best-match querying. In chapter 6, we describe the database foundations for preference-based queries and our approach to support them in a peer-to-peer context. Chapter 7 gives a summary and outlook[5].

[5]Chapters 3–6 discuss in detail the author's research conducted at L3S Research Center under the supervision of Prof. Dr. Wolfgang Nejdl on schema-based peer-to-peer systems and library networks, and are based the following publications:

Chapter 3	Design issues and challenges for RDF- and schema-based peer-to-peer systems. *SIGMOD Record*, September 2003, pp. 41-46. Co-authors: W. Nejdl, M. Sintek Schema-based peer-to-peer systems. In *Ralf Steinmetz and Klaus Wehrle (Eds.). Peer-to-Peer Systems and Applications.* Springer, 2005. Co-author: W. Nejdl
Chapter 4	Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proceedings of the 12th International World Wide Web Conference (WWW2003)*, Budapest, Hungary, May 2003. Co-authors: W. Nejdl, M.n Wolpers, C. Schmitz, M. Schlosser, I. Brunkhorst and A. Löser. Super-peer-based routing strategies for RDF-based peer-to-peer networks. *Web Semantics* 1 (2), Feb. 2004, pp. 137-240. Co-authors: W. Nejdl, M.n Wolpers, C. Schmitz, M. Schlosser, I. Brunkhorst and A. Löser. A simulation framework for schema-based query routing in P2P networks. In *Proceedings of the Workshop on Peer-to-Peer Computing and Databases, EDBT*, 2004 Co-author: U. Thaden.
Chapter 5	OAI-P2P: A peer-to-peer network for Open Archives. In *Proceedings of the Workshop on Dist. Comp. Architectures for Digital Libraries, ICPP*, 2002. Co-authors: B. Ahlborn, W. Nejdl. Towards a modification exchange language for distributed RDF repositories. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2002. Co-authors: W. Nejdl, B. Simon, J. Tane.
Chapter 6	Top-k query evaluation for schema-based peer-to-peer networks. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, 2004. Co-authors: W. Nejdl, U. Thaden, W.-T. Balke. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)* 2005. Co-authors: W.-T. Balke, W. Nejdl, U. Thaden. DL meets P2P — Distributed document retrieval based on classification and content. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005)*, Vienna, Austria, September 2005. Best paper award. Co-authors: W.-T. Balke, W. Nejdl, U. Thaden. Querying the Semantic Web with Preferences. In *Proceedings of the 5th International Semantic Web Conference (ISWC)*, 2006. Co-authors: J. Z. Pan, U. Thaden.

A complete publication list can be found in Appendix B.

Chapter 2

Foundations

In this chapter, we review the fundamental techniques and algorithms underlying SPNs. First and foremost this is database technology: SPNs can be seen as an evolution of distributed database systems. We also give an overview about Semantic Web concepts and standards for data representation, as these techniques are used for metadata representation and querying in OAI-P2P. Nowadays peer-to-peer networks are a research area in their own right, and we review the state of the art in this area as well.

2.1 Relational Databases

The relational model was invented in the 1970s as reaction to the issues of hierarchical databases, which had been prevalent at that time: in hierarchical databases, applications worked direct on the physical record structure and had to be adapted each time the record structure changed. This called for a decoupling of logical record view and physical storage layout. Additionally, data could only be structured in a tree-based fashion which is often inappropriate.

In 1970, Ted Codd presented an alternative approach, the relational model [39], with a very simple data organization: All data records are stored as tables, and links between records are modeled by using the key of an object as reference attribute (foreign key) in another table. With the implementation of System R [9], the first efficient relational database system, and the subsequent release of IBM DB/2, the relational model became the dominant data organization and storage principle.

The basic notion in the relational model is the relation, defined over a set of attributes. Let $A = \{a_1, a_2, \ldots, a_n\}$ be the set of attributes, and $A_1, A_2, \ldots, A_n$ the sets of allowed values for each attribute a_i. Then we can see a relation R as set of tuples:

$R = \{(v_1, v_2, \ldots, v_n) | v_i \in A_i\}$
We call a each tuple *database object*.

Such a relation can be viewed as *table*, where each attribute specifies a *column*, and each tuple is a *row*. The attribute set A in conjunction with the sets A_i of allowed value form the *table schema*. We denote set of all table schemas of a database *database schema*. The set A_i of all allowed values for attribute i is the attribute *domain*.

Codd proposed a relational algebra to express queries on a set of relations, consisting of union, set difference, Cartesian product, selection and projection. We can define union, intersection and difference on relations as on any sets:

$R_1 \cup R_2 = \{t | t \in R_1 \vee t \in R_2\}$
$R_1 \cap R_2 = \{t | t \in R_1 \wedge t \in R_2\}$
$R_1 - R_2 = \{t | t \in R_1 \wedge t \notin R_2\}$
These are only defined if R_1 and R_2 have the same table schema.

The Cartesian product of two relations R_1 and R_2 is

$$R_1 \times R_2 = \{(r_1, \ldots, r_m, s_1, \ldots, s_n) | (r_1, \ldots, r_m) \in R_1 \wedge (s_1, \ldots, s_n) \in R_2\}$$

In addition to classic set operators, two further operators are introduced to represent queries on relational databases, the *projection* and *selection* operator. The projection operator produces from the input a new relation with only a subset of the original attributes. We can see it as kind of vertical selections. For R defined as above, the projection on attributes $B_1, \ldots, B_k$ is defined as

$$\pi_{B_1,\ldots,B_k}(R) = \left\{(w_1, \ldots, w_k) | \exists t = (v_1, v_2, \ldots, v_n) \forall_{i=1}^{k} \exists j B_i = A_j \wedge w_i = v_j\right\}$$

While projection gives us a column subset, we can choose a row subset using the selection operator. Given a boolean function f on the tuples of R, selection is defined as

$$\sigma_f(R) = \{t | t \in R \wedge f(t)\}$$

If R_1 and R_2 share some attributes, the Cartesian product often is not what you want because the resulting relation contains duplicate attributes. This is where the join operator comes in. It also combines tuples from two relations, but only these which have equal values for all attributes in an intersection of R_1s and R_2s schemas. The natural join is just a shortcut for a combination of the basic operators. When A_{R_1} is the set of attributes of R_1, and A_{R_2} the set of attributes of R_2, then the natural join of R_1 and R_2 is

$$R_1 \bowtie R_2 = \pi_{A_{R_1} \cup (A_{R_2} \setminus A_{R_1})}(\sigma_c(R_1 \times R_2))$$
$$\text{with } c(t) = true \Leftrightarrow \forall A_i \in A_{R_1} \cup A_{R_2} : \; R_1.A_i(t) = R_2.A_i(t)$$

All modern RDBMS are based on relational *bag* algebra, not on set algebra as described above. This means, that duplicate tuples stay in the relation as distinct rows, and some laws pertaining to sets do not hold (e.g., the distributive law). To keep our model simple, we will continue

with set-based formalization as usual in database models; a generalization to bags is straightforward.

Another simplification of our model is that we do not take into account grouping and sorting operations (γ rsp. τ operator).

2.1.1 Query Plans

Any database system is a complex piece of software, and principles of software engineering need to be applied to to arrive at a robust and efficient system. For large systems, modularization is the key to a good software architecture, where the system is decomposed into encapsulated components with well defined interfaces. Such a modularization has been achieved early in the evolution of relational databases, in the seminal System R [9]. The separation of concerns proposed in this system is (with modifications) still in use in all major database products. We won't discuss the issue of modifications to the database (transaction management, logging, recovery, etc.), because this is out of scope of our work. However, the query processing approach can be seen as model for our context, too, and thus it is worth to look at it in detail. One of the many ideas in system R is the division of query processing into several phases, query parsing, query plan generation and plan execution (see Fig. 2.1.1). During query parsing, the query is checked for validity (syntactical well-formedness, existence of all referenced tables and columns, conformance of attribute types with used operators), and – if it is a valid query – transformed into a logical query plan. Now the crucial phase, query plan creation, starts. In this phase, the query is converted from a relational algebra representation into a physical query plan. In the first step, rewriting, the query is transformed into an optimized relational algebra expression; afterwards that expression is translated into a tree of physical database operations.

Both parser and rewriter/optimizer rely on the metadata about the database relations, foremost the database schema. All metadata is stored in a (logically) separate database, the *catalog*.

Let's review the query plan generation with a simple example. Suppose, we have the tables PUBLICATION, PERSON and AUTHOR, where the last one describes a persons authorship of publications:

PUBLICATION(ID, TITLE, DATE, LANGUAGE)
PERSON(ID, NAME, SURNAME)
AUTHOR(PUB_ID, PERSON_ID)

The standard language to access relational databases is SQL [34, 168]. If we want to find English translations of the works by Immanuel Kant, we could express that query in SQL as

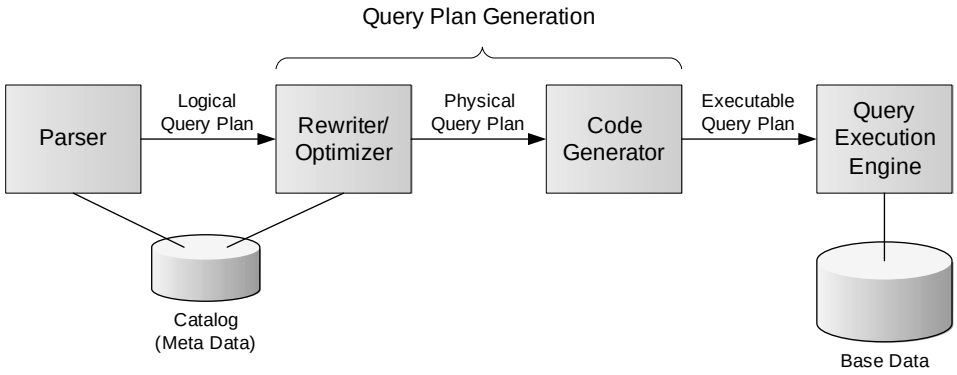


Figure 2.1: Query Processing Overview

```
SELECT PUBLICATION.ID, PUBLICATION.TITLE
FROM PUBLICATION, PERSON, AUTHOR
WHERE AUTHOR.PUB_ID = PUBLICATION.ID AND AUTHOR.PERSON_ID = PERSON.ID
    AND PERSON.NAME = 'Kant' AND PERSON.SURNAME = 'Immanuel'
    AND PUBLICATION.LANGUAGE='en'.
```

A straightforward logical query plan is to join all tables, then select according to the person constraints and finally project the title:

$\pi_{\text{ID, TITLE}}(\sigma_{\text{NAME='Kant'} \wedge \text{SURNAME='Immanuel'} \wedge \text{LANGUAGE='en'}}((\text{PERSON} \bowtie \text{AUTHOR}) \bowtie \text{DOC}))$

Obviously, this would require access to all objects of all three tables, a very inefficient query plan. A better solution is to perform selections based on the right person first, then join with the author and document tables, select with respect to the language constraint, and finally project id and title:

$\pi_{\text{ID, TITLE}}(\sigma_{\text{LANGUAGE='en'}}((\sigma_{\text{NAME='Kant'} \wedge \text{SURNAME='Immanuel'}}(\text{PERSON}) \bowtie \text{AUTHOR}) \bowtie \text{DOC}))$

How query plan optimization works is sketched later.

Physical query plan While the RDBMS interface is set (or, more correctly, bag) based, internally at some point the declarative style must finally be converted to an imperative one, because current processor instruction sets are based on the imperative paradigm. The physical query plan is the connection between the declarative and imperative world. It is produced by transforming the declarative logical query plan, but the implementation of each of it's operations is provided (by the database manufacturer) in an imperative style.

Modern physical database operators follow the iterator pattern [64]. Each operator provides an API for starting an iteration, and then requesting the next matching tuple. This style allows to plug together operators freely, because all provide the same access interface.

The most basic physical operation is the table scan which iterates over a table and delivers selected tuples. Suppose that an index for PERSON.NAME is available. Then a good physical

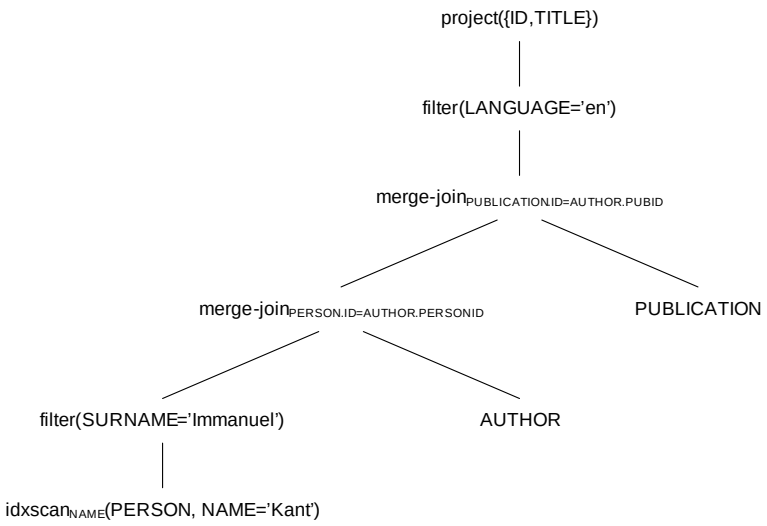


Figure 2.2: Physical Plan Example

plan would be the one shown in Fig. 2.2. Table scan operators such as scan and idxscan execute selection and projection. The scan operator directly iterates over table blocks, while idxscan uses an index to access only requested tuples.

Query Plan Optimization Logical as well as physical query plan are optimized according to estimated operator costs. For example, selections are pushed down as far as possible to reduce the number of accessed tuples. In general, for each operation, a processing cost is estimated, and the plan is constructed so that the complete query cost estimate is minimal. It is already a difficult task to estimate processing correct costs for a simple operator, because this may depend on many factors (table size, distribution of values within one column, size of input set from preceding operation). Cost estimation is based on constants determined for the underlying database engine and on statistics collected on the current database. These statistics include current relation sizes, actual cost of operators collected during previous query executions, etc. Due to the difficulty of correct cost estimation, heuristics are also taken into account when deciding between different alternatives.

Based on this estimates, algorithms have developed which search for the optimal query plan for a given query. One of the frequently used approaches is *dynamic programming*. In this algorithm, all possible plans are expanded in a breadth-first manner. Partial plans are deleted (pruned) as soon as it turns out that they can't become optimal anymore. The other plans are further expanded until an optimal complete plan is found. Dynamic Programming works well for joins up to 6 relations, but becomes too expensive for more joins due to the exponential complexity of the algorithm. It has be shown that finding the optimal query plan is

NP-complete in the number of joined tables.

Therefore, various heuristic strategies have been developed which try to focus on promising areas in the query plan search space instead of doing exhaustive search. The simplest one is the *greedy algorithm*, which builds only one plan (depths first search) based on heuristics. However, the outcome of this algorithm is often far from optimal. Other algorithms such as *simulated annealing* [77] or *iterative improvement* [175] also start with one or several plans created heuristically, but then go on to apply (random) modifications to find a better plan. It has been shown experimentally that for more than a few involved relations, such algorithms lead to nearly optimal plans with a substantially improved performance [93].

In today's databases the boundary between logical and physical query optimization has become more and more blurring, up to the point where only one integrated plan optimization process is used [65].

Summary The relational model provides a well-defined and proven foundation for modeling complex query processing algorithms. Query plans are an established device to separate the logical query level from physical, by transforming declarative queries to imperative-style execution instructions.

2.2 Distributed Databases

Often, related data is not stored in one central database, but distributed over several physical locations. For a client the optimal approach is access to that distributed context as if it would be a single database. Therefore, prototypes of distributed database systems have been developed very early, e.g., System R* [103] or Distributed Ingres [171].

With a homogeneous schema, distribution can occur in two different ways:

- Different relations or different attributes (columns) of one relation can be stored on different nodes. This is called *vertical* data distribution.
- Tuples of the same logical table can be stored on different nodes. This is referred to as *horizontal* distribution.

The case of distributed querying in the context of heterogeneous schemas is much more difficult, and an active research area in itself. In the late 90s, several research projects started to build solutions for loosely interconnecting data sources available via the Web. TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources, [136]) is one of the first such

systems for integration of semi-structured data from Web sources. It employs a mediator-wrapper architecture, where a central mediator takes care of query planning, while specialized wrappers translate (partial) queries to source-specific requests. TSIMMIS focuses on selection and projection, based on the argument that joins are unlikely in the Web context. Other data integration systems are Information Manifold [96], AmosII [80], Garlic [81], Disco [180] and Strudel [55] (the predecessor of Piazza, cf. 3.4.2).

In this thesis, we do not discuss the case of heterogeneous schemas and schema integration methods and focus only on query plans for the homogeneous case.

Distributed query plan Query plan creation for the distributed case works like in the local setting, but some new factors have to be considered. First, the query optimizer has to find out where the tables (or table fragments) referenced in the query are located. It is not necessary to maintain a complete catalog at this site; if only partial information is available, a partial plan can be created and forwarded to other nodes, where it is refined until the plan is complete [25]. The next step is to split the query into subqueries, according to the distribution of the data. The best way to model this is by introducing an additional pair of operators, *send* and *receive* [86]. The *receive* operator acts as a proxy and offers the same iterator interface as the other physical operators. The *send* operator sends result tuples of a local sub-query, triggered by requests from its *receive* sibling in a consumer-producer fashion. These operators encapsulate all communication issues so that all other operators don't need to distinguish between local and distributed evaluation.

The optimization algorithm can basically stay the same as in the central case, but needs to take additional factors into account. Cost factors as bandwidth and latency have to be involved into the cost estimation to achieve optimal plans. Also, the number of possible plans increases, because the send/receive operator pairs can be added at various positions in the plan. If more processing is done at the distributed nodes (*query shipping*), the amount of data sent is smaller, but the processing costs at the affected nodes might be higher. If more 'raw' data is transferred to the coordinating node instead (*data shipping*), and the query execution takes place there, then this involves typically higher communication costs, but can reduce processing efforts. Instead of pure query or data shipping, intermediate approaches can be used; this often combines the advantages of both approaches [57]. The picture becomes even more complex if caching is introduced at the coordinating node. As query plan enumeration is exponential, exhaustive visit of the complete query plan search space is out of question for distributed queries. Therefore, new heuristics (such as the ones sketched above) have to be introduced for selecting promising plans. Finally, more trade-offs have to be made, e.g. shortest response time vs. shortest overall query processing effort.

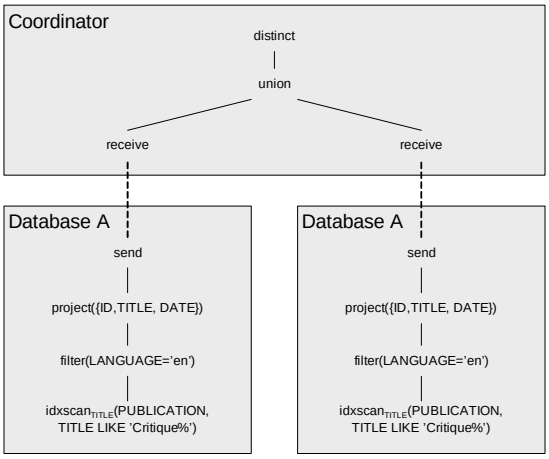


Figure 2.3: Distributed Query Plan Example

As an example, let's take a look at query plans for horizontal distribution. To reconstruct a horizontally fragmented relation, it is just required to union the fragments from all nodes which provide this relation. Figure 2.3 shows such a case for a query for all publications in English with a title starting with 'Critique'. Due to the distribution, new opportunities and requirements for optimization occur: One important optimization is to determine nodes which will produce empty relations as result, and remove them from the plan: if relation constraints for nodes (e.g., value ranges) are known, then these can be compared to the selection constraint, and all nodes where they are contradicting can be omitted from processing. Another common optimization is pushing down selections and projections to the involved nodes.

If the horizontally fragmented relation is to be joined with another relation, it is often advantageous to push down the join operator. Suppose we want to join relations R and S, and R is partitioned into two horizontal fragments R_1 and R_2. Then, the join can be computed as $(R_1 \cup R_2) \bowtie S = (R_1 \bowtie S) \cup (R_2 \bowtie S)$. For example, in the library catalog context we can safely assume that a site provides the required person tuples for all authors of its documents. In this case, we can execute partial joins at all nodes first and union the results afterwards.

To merge a vertical distribution, a join on the table key needs to be performed. If the query is a projection, we can optimize the plan by leaving out nodes which do not store projected attributes.

Federated databases Often, data sources in a distributed setting are highly autonomous and not willing to integrate so tightly that distributed query plans are possible. Such a scenario is called database federation [163]. In this case, the coordinating instance maintains a *federated schema* and mappings to the participants schemas. Queries are posed in the federated schema. However, in contrast to tightly coupled systems, the federated database management

system needs to convert the partial query plans back to partial queries that are accepted by the respective database. Then, it sends the resulting queries via the respective offered query interfaces.

To achieve a flexible system architecture, the translation of query plan fragments into system-specific requests can be delegated to backend wrappers (e.g., in [136]). This allows for easy extension of the system, when new types of data sources have to be integrated. This wrapper architecture has been used as model for our prototype architecture (cf. 5).

Large-scale distributed database systems Most approaches in distributed databases aim at integration of comparatively few database nodes. Two more recent systems tackle the problem of interconnecting a larger amount of nodes, Mariposa [172] and ObjectGlobe [25].

Mariposa uses a microeconomic model for cost optimization in the query planning process. Here, a client allocates a budget together with his query, and the (central) query planner asks data sources for bids on query operator execution. The cheapest provider is selected. This approach increases scalability of the overall system, compared to purely central cost estimation. However, a solution to overall currency management has not been devised. Thus, the approach is not easily applicable to open networks.

The goal of ObjectGlobe is to fulfill the vision of the *"Internet as a global database"* [102]. It can also be viewed as distributed query processor, where a query plan is devised and executed based on schema information. ObjectGlobe assumes a global catalog provider where all schema information is stored [1]. Nodes can act as data providers (providing information), cycle providers (providing processing power), and function providers (providing any kind of additional functions, such as aggregation or transformation functions). All providers can join and leave dynamically, but have to register with their self-description at the central catalog instance. In contrast to traditional mediator-based systems, not only sub-queries are delegated to data providers, but the whole query execution is distributed. As mentioned, participating nodes can provide data as well as computing power, and the query plan is devised to optimize query processing performance across all nodes. The plan is executed using the iterator model described in 2.1.

While these systems advanced the scalability of distributed query processing substantially, they are still far away from internet-scale dimensions. For example, Mariposa aims at integration of *"1000 sites or more"*([172]). The evaluations performed in these and other systems [136, 96, 80, 81, 180, 55] typically included less than a dozen sites. See Section 3.4 for an overview of peer-to-peer systems for distributed query processing.

[1] This catalog can be physically distributed over a cluster of catalog provider nodes

Summary Traditional distributed database techniques aim at tight integration of participating data sources. This has been achieved by evolving the query plan approach to facilitate distributed query plans. Distributed query plans are created by a central coordinating instance. Thus, these approaches can't be applied directly for the creation of decentralized networks of independent data sources. Besides, the distributed query planning and execution algorithms don not scale to large numbers of nodes. However, they build an important starting point for most SPN approaches, as we will show in Chapter 3.

2.3 Semantic Web

The Semantic Web initiative is a reaction on two factors regarding the evolution of the Web: on the one hand, it proved to be a very successful medium to disseminate information. On the other hand, this information, while often in the form of data, is very hard to access in an automated fashion. For example, nearly every large library provides an online version of its catalog, but these Web interfaces can't be easily used as software services, because their input and output formats are optimized for human users, and difficult to interpret by machines. This is where the vision of the Semantic Web kicks in: According to that vision, whenever some structured, machine-processable information is available, it is provided along with the human-readable context on the Web. Reasoning processes use this additional information (optionally together with the content itself) to find resources matching users' information needs. Initially, the initiative focused on representation standards for such information, but recently generic information request facilities also came into the focus of Semantic Web standardization (see 2.3.2).

Some Semantic Web pioneers use the term 'meaning' to characterize their vision. For example, in [18] Berners-Lee et al. state: "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". This terminology might be a bit misleading; if at all, computers have a completely different way of 'understanding' a 'meaning' than humans. Besides, it is not so much the 'well-defined meaning', which is missing on today's Web pages. On the contrary, high-quality Web information sources provide their information in a way very easily understandable by humans. What makes it difficult to devise automated processes for information extraction is not missing 'meaning', but missing (or non-uniform, or not formally-defined) regular format or structure.

Semantic Web Tower Accordingly, what the Semantic Web provides so far are standardized means to formally structure and represent information. These standards are meant to be layers

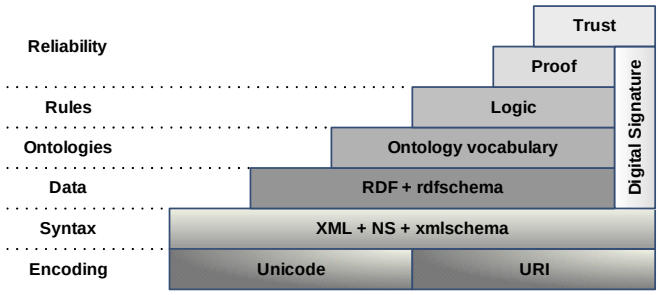


Figure 2.4: Semantic Web Tower

in a 'Semantic Web tower', as shown in Figure 2.4 (adapted from [16]). The tower consists of the following layers:

- **Lexical Layer** This layer is shared with the current Web and provides basics for information and link encoding, namely using Unicode character encoding and URIs as unique resource ids.
- **Syntactical Layer** The syntactical layer provides the foundations for RDF syntax: XML as machine-processable format, namespaces to abbreviate URI fragments and XML Schema to attach data types to raw strings.
- **Data Layer** The data layer consists of the Resource Description Format (RDF), a graph-based format to represent simple statements, and RDF Schema (RDFS), consisting of some basic constructs to create schemas for RDF-encoded data (see 2.3.1).
- **Ontology Layer** RDFS provides only limited means to specify relations between classes and between properties (actually only subsumption). The ontology layer significantly increases expressivity for specification of such relations. It uses Description Logic [10] as formalism to specify full-fledged ontologies. The ontology layer has recently been standardized; its ontology description language OWL (Ontology Web Language) comes in three flavors: OWL Lite, OWL-DL and OWL Full [48]. These flavors differ in language expressivity, and accordingly, in the degree of computability. OWL Lite is decidable in polynomial time, but not much more expressive than RDFS. OWL DL offers the complete Description Logic expressivity, at the price of NP-completeness. OWL Full is even undecidable and thus not of much practical use.
- **Rules Layer** Not every association between statements can be described as part of a (DL) ontology. Therefore, a separate layer is envisioned which facilitates the addition of rules to knowledge bases which allow to derive further information from existing one. Rule based extensions are currently discussed in a W3C working group [153].
- **Reliability Layer** When it comes to automated exchange of information on the Web, the

ability to assess the quality and reliability of information sources is crucial. Currently, it is unclear how to reach that goal. Proof and Trust seem to be rather placeholders denoting unresolved problems than actual layers providing solutions.

This tower model has recently spawned a lot of discussion. During design of the Ontology layer, it turned out that especially RDFS is rather a burden than a helpful foundation for an ontology description language (see [134] for a detailed discussion). For the upper levels there is no standard available yet. Interestingly, we already see similar problems regarding the interplay between ontologies and rules as before between RDFS and ontologies: it seems that the ontology layer expressivity has to be restricted (in this case, probably significantly) to ensure proper interplay between ontologies and rules without losing decidability [73].

The work presented in this thesis does not support interconnection of ontology-based information sources, but is limited to RDF and RDFS. Therefore, we will only describe these standards in more detail, and won't consider OWL and higher Semantic Web layers in the following.

2.3.1 RDF and RDFS

In RDF, all information is encoded as (sets of) statements. Each statement consists of a subject, a predicate and an object (always in that order). Subjects are uniquely identified by URIs. For example, to say that the book "Kritik der reinen Vernunft" was written by Immanuel Kant in 1781, we first need to assign it a URI, let's say http://books.org/kritik. Now, we encode our knowledge in statements, first in an informal syntax:

http://books.org/kritik *has_title* "Kritik der reinen Vernunft".
http://books.org/kritik *has_creator* "Immanuel Kant".
http://books.org/kritik *has_publication_date* "1781".

Items identified by URI are called *resources*, and data items (like all objects of the sample statements) *literals*.

So far, we have gained little, because the predicate part of our statements has been chosen in an ad-hoc fashion. For example, another source may have chosen *has_author* to denote the information in the second statement. What we need is a description schema on which all contributing information sources agree, at least to some degree. As mentioned in section 1.1, the Dublin Core Initiative provides exactly such a standardized vocabulary. It is available in several technical formats (bindings), among them an RDF binding. In RDF, the 'verbs' used to connect subject and object at the predicate position are called properties. To avoid name clashes, property names consist of a namespace and a name. The namespace is a URI, but a shorthand can be introduced using XML namespace syntax and semantics. We will use `dc` as shorthand for the Dublin Core vocabulary and `books` as shorthand for ``` ``http://books.org/'' ```; then

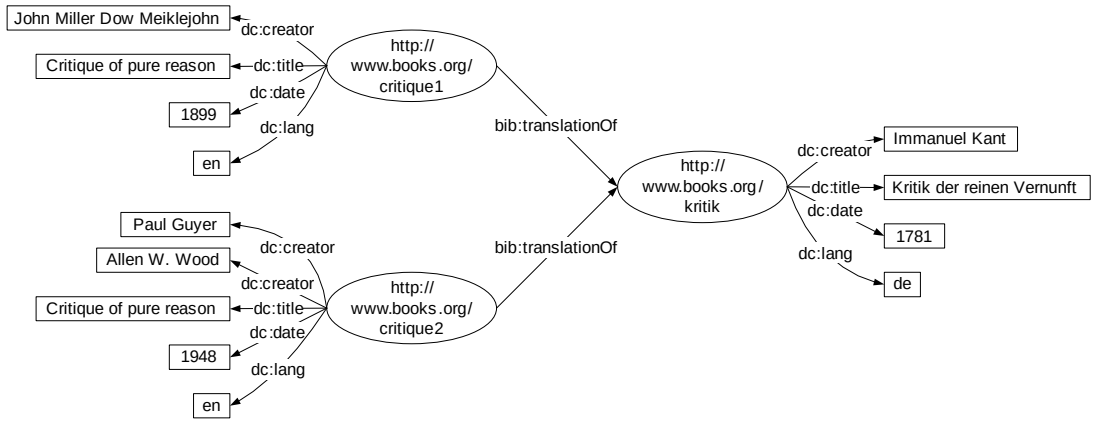


Figure 2.5: Sample Knowledge Graph of Book Relations

we can write our example statements as follows (in N3 notation [15]):

```
namespace dc "http://purl.org/dc/elements/1.1/".
namespace books "http://books.org/".
books:kritik dc:title "Kritik der reinen Vernunft".
books:kritik dc:creator "Immanuel Kant".
books:kritik dc:date "1781".
```

There is also a standard XML format for RDF statements [13].

To visualize RDF statements as knowledge graphs, a common notation has been established for RDF: Resources are depicted as ellipses, literals as rectangles, and properties as arrows connecting resources and literals. The (sub-)graph visualizing the “Kritik der reinen Vernunft” metadata is shown on the right hand of of Fig. 2.5.

With this simple example, one may wonder why a new data format was required at all, because such information is expressible in XML equally well. But the advantages of RDF come into play if we start to interconnect basic resource descriptions, thus creating larger graphs of knowledge. For example, we can include relations between books into our model, as shown in Figure 2.5.

Now, we can pose questions to our knowledge base such as *Give me all translations of Kant’s ‘Kritik’ into English*[2]. Note that the nodes and edges in that graph do not necessarily need to come from the same source. For example, we could imagine one source specializing on philosophers in the 18th century, another dedicated to Kant’s work only, and a third specialized on translations of German works into English.

[2]Current library metadata sets do not provide the depicted *translationOf* relation; however, it could be derived from the *Uniform Title* attribute (cf. Figure 1.1).

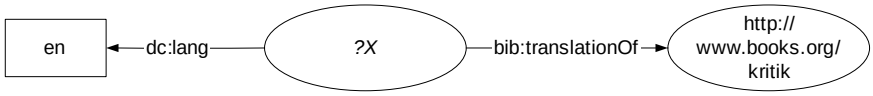


Figure 2.6: Graph pattern for query *Give me all translations of Kant's 'Kritik' into English*

2.3.2 Semantic Web Query Languages

A data model without the ability to pose queries against it is quite useless. Therefore, it is no wonder that a lot of RDF/S query languages have been proposed. [139] lists 20 different Semantic Web query languages, and is still not complete. We can distinguish *graph pattern matching*, *rule-based* and *XML-derived languages*.

Graph pattern matching languages In these languages, the condition is always a graph pattern, where some variables are free. Figure 2.6 shows such a pattern for the sample query above (where *?X* denotes a free variable). As a result, either a list of tuples with valid variable bindings is returned (as in SQL), or these bindings are used to construct a new graph from given construction template.

The most prominent representatives of this group are RDQL [72], RQL [83] and the upcoming standard SPARQL (current draft is [140]). SPARQL (and its predecessor RDQL) are working on the raw RDF graph and do not by default take RDFS semantics, e.g., subsumption relationships into account. However, RDFS or OWL semantics can be introduced by complementing the RDF graph with all statements entailed by the chosen semantics. RQL is RDFS-aware, and even provides semantic sugar to deal with conditions on schema level.

The following example illustrates the differences. To get all paperbacks with title starting with "Critique", you need to write the following in SPARQL:

```
SELECT ?X WHERE {
   ?X rdf:type book:Paperback.
   ?X dc:title ?T.
   FILTER (REGEX(?T, "Critique.*"))}
```

while in RQL you can select from a set of class instances:

```
SELECT X FROM {X;b:Paperback} dc:title {T} WHERE
   T LIKE "Critique*".
```

The latter query will also find books of subtype of book:Paperback, while the former matches only resource of exact type book:Paperback.

On the other hand, SPARQL offers an extensive set of query modifiers beside the pure graph pattern matching expressions. FILTER constructs can be used to impose additional constraints,

ORDER-BY specifies the output order of solutions, and LIMIT can be used to restrict the number of solutions returned. By default solutions are returned as tuples of variable bindings. But it is also possible to let a query construct a new (sub-)graph for each solution, using the CONSTRUCT expression, making it possible to nest queries.

When compared to the relational model, it turns out that RDF subgraph matching basically can be expressed in relational terms as well, by representing RDF statements as rows in a statement table, such as `STATEMENTS(SUBJECT, PREDICATE, OBJECT)`. Then, a graph matching statement can be translated as self-join on this table. For example, the sample SPARQL query above would be translated to

```
SELECT T1.SUBJECT AS X FROM STATEMENTS T1, STATEMENTS T2 WHERE
  T1.PREDICATE = 'rdf:type' AND T1.OBJECT = 'book:Paperback' AND
  T2.PREDICATE = 'dc:title' AND T2.OBJECT LIKE 'Critique%' AND
  T1.SUBJECT = T2.SUBJECT
```

[45] gives a translation of most SPARQL constructs into relational query plans, based on a statements table. He also points out the few exceptions where SPARQL semantics is not compatible with the relational model, e.g., in the handling of missing rsp. null values. A similar approach has also been taken by Oracle to add RDF support to their database product [38].

SPARQL is nearing its standardization, and will very likely become the dominant Semantic Web graph pattern matching language.

Deductive Languages Languages in this group are logic based: they view RDF statements as facts, and try to prove clauses based on these facts. deductive languages provide graph pattern matching capabilities as well, but extend query expressivity, because they allow to specify additional rules. When succeeding, they also return all possible bindings for free variables. Members of this group are N3QL [17], TRIPLE [166], RuleML [20], and QEL (see 5.3). To illustrate the way deductive languages work we use TRIPLE as example. TRIPLE offers (Horn-) logic expressions including existential and universal quantifiers. The sample query shown in SPARQL and RQL is expressed in TRIPLE as:

```
FORALL B,T result(B) <--
  B[rdf:type --> book:Paperback;
  dc:title --> T] AND like(T, "Critique.*") @booksModel
```

Note that the model the query is evaluated against has to be specified explicitly; in this case, it is `@booksModel`. While TRIPLE doesn't adhere to RDFS semantics by default, they can be specified by adding rules to the original query. For example, to add all type statements inferable by subclass declarations, we can use the following rule:

```
FORALL M @subsumption(M) {
  FORALL O,T O[type-->T] <--
    EXISTS S (S[rdfs:subClassOf-->T] AND O[rdf:type-->S]).
}
```

This query creates a new model `@subsumption(M)` including the additional statements from a given model M. For the complete RDFS rule set see [119]. Queries can be chained using the result model of one query as input for the next one. For example, to take all subclass relationships into account, the sample query would just evaluate against `@subsumption(booksModel)` instead of `@booksModel`.

XML-based Languages As there is an XML representation of RDF, it seems a good idea to reuse XML query languages as foundation for an RDF query language. This has been tried in two different ways: [148] have proposed to use XQuery without change. This poses the following problem: any RDF graph can be represented in RDF/XML in different ways, and an XQuery query would evaluate differently for each representation. To overcome this issue, a canonical, unambiguous RDF/XML representation is introduced. This approach is called *Syntactic Web*, because no semantics are supported; it's just the RDF graph that is evaluated. However, graph extension by entailment could be supported as additional step before query evaluation. A similar approach is taken by Excerpt [28], but with another canonical mapping from an RDF graph to XML. The disadvantage of XML-based querying of RDF/S is that the result set is an XML structure, not an RDF graph. This makes chaining queries difficult.

Summary The lower Semantic Web layers, up to the Data layer, are already stable and provide two important facilities:

- a unique identification scheme for resources, enabling annotation of the same items from different, uncoordinated information sources.
- schema specification support, enabling the standardization, but also flexible extension, of data representation formats.

For the higher layers, it seems that the development has to continue further to arrive at solutions usable in large scale distributed networks.

The same picture is recognizable in Semantic Web query languages: Graph pattern matching languages are already mature and a standard is approaching (SPARQL). Logic-based languages are significantly more expressive, but still research projects. While a unification of RDF and XML is desirable, current proposals for XML-based RDF query languages do not fulfil the expectations, because they don't take significant characteristics of RDF and RDFS into account.

2.4 Peer-to-Peer Networks

Peer-to-peer (P2P) networks are determined by three characteristics, self-organization, symmetric communication and distributed control [146]:

- **Self-Organization** In contrast to other distributed systems, P2P networks are not administered. Their underlying algorithms take care of all required maintenance. Specifically, P2P networks allow free join of new peers and departure of current ones, and adapt network connections accordingly.
- **Symmetric Communication** While in traditional client-server systems a node is either server or client, in P2P networks all nodes play server *and* client roles.
- **Distributed Control** P2P systems do not have a single point of control, such as a central naming directory or access control point. The control is distributed among the nodes, so that no node becomes a single point of failure. Whenever a node fails, its share of the network control is taken over by other nodes.

P2P systems exhibiting not all of these characteristics are called *hybrid P2P systems*. We will review P2P search techniques for basic queries here (i.e., queries on one attribute only), and show how these are applied to build SPNs in Chapter 3.

The first P2P networks have been called *unstructured*: new peers connected in an arbitrary fashion to existing ones to join the network, resulting in a random topology. The most prominent representative of these systems was Gnutella, protocol version 0.4 [61]. In these systems, queries are flooded: they are recursively broadcast to all connected peers, until a predefined horizon, i.e., a specific number of node hops from the query originator, is reached. Objects stored outside of this horizon are not considered during query evaluation. All query messages are distributed in the same way, regardless of their content.

The next step was the development of Distributed Hash Table (DHT) based P2P networks. These networks assign a key to each object provided by a peer. The only operation supported is lookup by this key. But this restriction is more than compensated by the advantages of DHTs: Any objects within the network matching the lookup key are guaranteed to be found (not taking large-scale network failures into account), and query routing is very efficient. To achieve these characteristics, DHTs rely on organizing the nodes into a highly regular graph. Consequently, these systems have been classified as *structured* P2P networks.

While this distinction into structured and unstructured networks already has some tradition, it nowadays becomes less and less useful, for the following reasons:

- While query distribution via flooding doesn't require an underlying regular topology,

it can benefit from it [32, 157]. This shows that query routing method and network topology should be viewed as conceptually different (although related) aspects.

- It has been turned out that in some contexts a combined DHT/Flooding solution can outperform pure flooding as well as a pure DHT [104].
- Recent approaches [118, 155] use randomly formed topologies which however provably converge against a graph with regular characteristics over time (e.g., small-world characteristic). These topologies are neither structured nor unstructured in the usual sense.

Therefore, a better approach is to classify existing P2P approaches according to their indexing method instead of their topology type, as in [146]. This yields three categories, local, central, and distributed indexing. We describe these types in the following subsections.

2.4.1 Local Indexing

Local indexing systems do not have any distributed index at all (as first Gnutella versions). Query routing here comes in two flavors, flooding [61, 189] and random walks [6, 108]. In flooding, a query is copied during propagation, and the copies are forwarded in parallel, while in random walks the query message is passed as token from peer to peer until a hit is found. No system using random walks has been deployed on larger scale yet. While several attempts have been made to improve routing in such systems, e.g., by using success heuristics [189] or by directing preferably to higher degree nodes [6], it is agreed nowadays that pure local index systems don't scale well [147]. [104] found that search success rates are high for popular and thus highly replicated items, but rare items often are not found although they are provided within the network. Increasing the search horizon improves success rates, but also imposes a higher message load on the network, and consequently reduces scalability.

2.4.2 Central Indexing

A centralized index contradicts the P2P characteristics given above (no central control), and therefore systems using such an index fall in the class of hybrid P2P systems. At first sight, a central index seems completely unattractive, due to the load imposed on the central node. But Napster showed that one node can easily cope with this load if only the index access, and not the data access is routed via the center [123, 154].

Probably due to the violation of P2P spirit no significant research of such systems has been pursued. However, the idea of splitting index responsibility from data provision responsibility has stayed alive, and has lead to the development of super-peer systems (see below).

2.4.3 Distributed Indexing

Most current P2P approaches fall into the distributed indexing category. Here, no node in the network has all the indexing information, but every node has some part of it, sufficient to forward query messages in the right direction(s), according to their content.

Distributed Hash Tables The most prevalent distributed indexing algorithms are distributed hash tables (DHTs). DHTs basically support search by key only. Each resource stored in the network gets a key assigned, and can subsequently be looked up via this key. DHTs work by assigning a part of the key space to each node and creating a structured connection between all nodes. The topologies used are highly symmetric, and make each node a root in a spanning tree covering the whole network. Therefore, for each key there is a unique route from the lookup originator to the peer responsible for that key. This approach makes routing key-based queries very efficient: On average, for a network of size n, a query reaches the node responsible for the key in $O(\log n)$ steps.

The dominant topology for DHTs is derived from Plaxton trees [138]. Systems such as Chord [170], Pastry [151] and Oceanstore [88] rely on this structure. Other graph structures used for P2P topologies are de Bruijn graphs [82, 122] and Butterflies [110].

A different approach is used by the Content-Addressable Network (CAN, [141]). Here, a d-dimensional torus, forming a Cartesian coordinate space, is used as underlying structure. Each peer is assigned a d-dimensional rectangle zone within this space. Lookup-wise, CAN doesn't perform as well as other DHTs: its average hop-count until the lookup succeeds is not logarithmic in the network size, but $O(d \cdot n^{1/d})$.

In Symphony [111] a topology with small-world characteristics is created. As in the other approaches, each node is assigned a key space interval. The advantage of Symphony is that only a constant number k of *long distance links* has to be maintained by each peer, independently of the network size. Nodes also form a ring, and select their long distance links according to a harmonic distribution. This yields an average latency of $O(\log^2 n/k)$ hops.

P-Grid [1] uses a topology derived from the trie structure. As with other DHT approaches, the key space is partitioned among peers. Here, the partition is organized according to key prefixes: each node is responsible for a specific prefix. To facilitate load balancing, the prefix length may vary, so that densely populated key intervals can be distributed among more peers than sparsely ones. For very skewed distributions this leads to larger routing tables (in the worst case linear with network size), because each peer maintains a link to all complementary prefixes. On the other hand, the average latency stays $O(\log n)$ even for skewed distributions. P-Grid does not require a random hash function, because of its ability to cope with skewed

key distributions. When using an order-preserving hash function, range queries can be easily evaluated, by first routing the query to the left boundary of the interval, and then forwarding it along the successor link each node maintains, until the right boundary is reached. P-Grid has another property which distinguishes it from other DHT approaches: it provides a very efficient way to construct the underlying trie, where the key space is partitioned incrementally in parallel among the nodes, based on random encounters between them [4].

Filtered Flooding For key-based search, DHTs are well established and efficient solutions. However, other search types such as keyword search or even complex queries can't be easily supported by DHTs. Therefore, most systems allowing for complex queries use flooding as basic query distribution approach, but try to restrict destinations based on query and index information. In other words, the index acts as filter on the underlying topology. This kind of index can either be built in advance, or based on query statistics. In the first case, peers post content summaries which are stored in indexes, and queries are matched against these summaries [68, 85, 22, 126]. A query is only forwarded to a peer if its self-description matches against the query. In the latter case, queries first are distributed, and indexes are built based on peers responses [12]. That way, indexes are built as byproduct of query processing, therefore no index maintenance messages are required.

Short-cut networks Going one step further, the topology itself can be optimized by modifying links according to query and response statistics [105, 159]. Each peer tries to diminish its distance to the peers which have resources most frequently requested by this peer. Such networks are called short-cut networks, because they rely on creation of new short-cuts which replace detours in the networks.

Interestingly, specific reconnection strategies can lead to the emergence of regular topologies, although not predefined by the network algorithms [159]. This is characteristic for self-organizing systems in other areas (like biology) too, and seems to be a promising middle way between unstructured networks and networks with predefined regular topology.

Super-Peer networks In each network, the connected computers do have different capabilities regarding processing power, storage, bandwidth, availability, etc. Thus, to treat all peers equally would result in overloading small peers while not exploiting the capabilities of the more powerful ones. As discussed in [190], exploiting the different capabilities in a P2P network can lead to an efficient network architecture, where powerful and highly available peers, called *super-peers*, form a network backbone to which all other peers connect. These super-peers take over specific responsibilities, e.g., for query routing and schema mediation. Simple

peers don't form direct connections to other simple peers, but connect to a super-peer instead, and all messages are distributed via the super-peer backbone.

Two major file-sharing protocols, FastTrack[3] and Gnutella 0.6 [62], also use this approach. Both protocols assign the responsibility for indexing and query processing to their super-peers, called *supernodes* and *ultrapeers*, respectively[4].

Super-peer-based P2P 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. The last step can sometimes be avoided if the super-peers additionally cache data from their connected peers. The super-peer backbone can be organized according to any of the topologies mentioned above. Super-peer routing is usually based on different kinds of indexing and routing tables, as discussed in [43] and [126]. The super-peer approach is also used in SPQR (see Chapter 4).

Summary The earlier distinction between structured and unstructured networks is blurring, and P2P systems are much more characterized by their indexing approach than by their topology. Two indexing methods are predominant, DHTs and filtered flooding, which have complementary advantages and disadvantages. The predetermined structure of DHTs allows for more efficient query distribution in a lookup-based network, because each peer 'knows' the network structure and can forward queries just in the right direction. But this does only work if the data is distributed among the peers according to the anticipated search strategy. Also, it requires the restriction of query complexity. In filtered flooding systems, peers do not know exactly in which direction to send a query. Therefore, requests have to be spread further within the network than in DHTs, thus decreasing routing efficiency. Short-cut network approaches can reduce this difference, but filtered flooding will probably never reach as efficient query routing as DHTs. On the other hand, requests can take more or less any form, as long as each peer is able to match its resources against the request locally. That makes filtered flooding more suitable to complex query processing. Both methods can benefit from the introduction of a super-peer level which distributes the control share to the more powerful and reliable peers only, thus increasing network stability and avoiding overload of less capable peers.

[3]The Fasttrack protocol, developed by Sharman Networks Ltd, and used in the Kazaa network, is unpublished. However, an incomplete protocol description acquired by network traffic analysis is given in [70].

[4]The third prominent file-sharing protocol, BitTorrent, is a hybrid system that uses central indexing (via so-called tracker nodes), and exploits P2P only for the actual file download

Chapter 3

Design Dimensions of Schema-Based Peer-to-Peer Networks

When looking at the developments in P2P research in the past years, we can identify two stages: In the first stage, a lot of effort has been put into refining topologies and query routing functionalities for one-dimensional search or lookup, i.e., search based on one attribute only. The issue of multi-dimensional lookup and support for more expressive queries has been tackled in the second stage, and nowadays we see several systems exploring P2P infrastructures for more expressive queries. According to the query expressivity they support, current P2P systems can be distinguished into key-based, keyword-based and schema-based systems [46].

A complementary trend can be observed in the database area: here, approaches have evolved from centralized systems toward a higher degree of distribution. Traditional distributed databases have until recently assumed a high degree of central control for query processing, but we currently see first explorations toward true peer-to-peer data management infrastructures which have all characteristics of P2P systems, i.e.,*self-organization*, *symmetric communication*, and *distributed control of query evaluation processes* (cf. 2.4). In this view, schema-based P2P networks (SPNs) are the point where these two directions of research meet [66], as shown in Figure 3.1[1].

Recently, P2P systems with database-like querying capabilities have also been called Peer Data Management Systems (PDMS) ([68, 71]). In our opinion, current systems are not sufficiently capable to justify this terminology. Today, query capabilities of all P2P based distributed database systems are pretty limited in comparison to current DBMSs, and their 'management' functions (transactions with ACID guarantees, access control, fault tolerance, etc.) are not supported at all in existing implementations. It is also doubtful if fully decentralized data management – in contrast to decentralized query processing – has any practical application. Therefore,

[1] This figure is not meant to be complete, we have just depicted a few systems for illustration purposes.

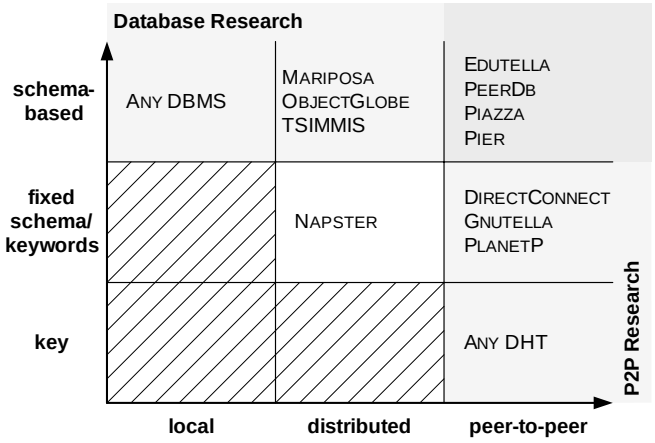


Figure 3.1: Schema Capabilities and Distribution

in this thesis we will stick to the term schema-based P2P network introduced in [124].

As we have seen, highly efficient algorithms for one-dimensional lookup, all of them belonging to the DHT family, have been devised (cf. 2.4.3). However, the lookup issue becomes much more complicated if we want to support item retrieval according to arbitrary (many) attributes, as it is common in databases. We can't just employ the approaches for one-dimensional lookup, because that would at least require the setup of a separate overlay network for each attribute which is already prohibitively expensive. Other issues arise when applying distributed database techniques to a P2P context. For example, no node has complete information about database schemas on all nodes, therefore mediator-based approaches can't be applied directly. Another issue is scalability: query planning in current distributed database systems only scales to at most hundreds of nodes (cf. 2.2).

It seems unrealistic to hope for an approach which works optimal in every respect. Instead, we have to make design decisions which will always involve trade-offs, e.g. between query expressivity and query processing efficiency. In this chapter, we explore the design space for SPNs, point out the trade-offs, and finally describe and classify existing systems.

3.1 Network Properties

The way the P2P network is organized and used to locate and access information is an important aspect of every data-sharing P2P system. [46] identify the following aspects with respect to the organization of nodes and data in such networks.

3.1.1 Data Placement

The data placement aspect 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. In a peer-to-peer system it seems most natural to store information at the node which is controlled by the information owner. 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. Also, the owner doesn't have to rely on availability of other peers responsible for fragments of its information. In the owner-based placement approach the network is only used to increase access to the information.

The complementary model is when peers do not only cooperate in searching information, but already in storing the information. Then the network as a whole works 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 strategies are employed in current systems. DHT-based SPNs such as PIER [75] and RDFPeers [30] distribute data according to its key, while Piazza [68], Chatty Web [2] and PeerDB [129] keep all data at the owner peer (see 3.4 for a detailed description of these systems). Gribble et al. [66] see the question of data distribution vs. query distribution as core research challenge in SPNs.

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 query evaluation performance, it also increases maintenance costs and reduces the owner's control of information.

3.1.2 Topology and Routing

As we have seen in section 2.4, diverse approaches to organize peers into a network overlay topology exist. It is agreed that for one-dimensional key-based lookup DHTs offer superior characteristics over other solutions. However, for SPNs the matter is not clear, because support for multi-dimensional queries is required. Current approaches can be sorted in the following categories:

- **Multidimensional DHTs** Here, the idea is to create separate overlays for each attribute. Representatives for this approach are RDFPeers and PIER. The advantage is that all accesses still require the guaranteed number of hops, typically $O(\log n)$. This allows for efficient query execution as long as not too many attributes are touched. Topology

maintenance and data insertion are more costly, but only by a constant factor (number of attributes). When using a cryptographic hash, such as in PIER, range queries are not supported. But with order-preserving hash functions (used by RDFPeers), range queries can be evaluated by starting at the peer responsible for the lower bound, and forwarding a query to a peer's successors until the upper bound is reached. Depending on the range size this can degrade to query flooding.

- **Unstructured Networks** Unstructured networks don't face the problem of creating index structures for all attributes. Instead, they match the schema information of peers and of the query to find promising routing directions for a query (filtered flooding). This approach is used by Piazza and Chatty Web.

- **Short-Cut Networks** Short-cut systems also do not impose a graph structure on the peer network. However, they try to optimize connections (find short-cuts) based on the results of past query evaluations. Although not constructed in the strict sense, this often results in specific graph characteristics, e.g., small-world networks. PeerDB, REMINDIN' [178] and INGA [105] are short-cut networks.

- **Super-peer Networks** Super-peer networks work in a very similar fashion as mediators in distributed databases. In these networks the responsibility for indexing (and subsequent query routing) is separated from the responsibility for data provisioning: A super-peer backbone collects peer self-descriptions and maintains indices accordingly. Queries are always forwarded to a super-peer, and distributed within the super-peer backbone. All super-peers which get the query forward it to their matching peers. Note that 'super-peer' and 'data provider' are just different roles which can be played by the same physical peer. The approach proposed in this thesis falls in the class of super-peer networks. Other schema-based super-peer systems are XPeer [156], SQPeer [85] and TOPICS [106][2]. The latter uses ontologies to categorize information: 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.

[2] SQPeer and TOPICS are actually inspired by the approach presented here

3.2 Data Storage and Access

3.2.1 Data Model

The data model used to store information is tightly connected to the choice of query language. Many data models have been proposed for storing data and we are not able to discuss them all in detail. We rather want to mention some basic distinctions with respect to the data model that influences the ability of the system. The most basic way of storing data is in terms of a fixed, standardized attribute set that is used across the whole system[3]. Despite the obvious limitations, fixed schema approaches are observed in common file-sharing peer-to-peer systems because this eliminates the problem of schema interoperability. Representatives for this approach from research are the structured networks MAAN [31] and Mercury [19], which both have a simple tuple data model. One step more complex is the relational model, the prevailing model for current non-distributed databases. Of course, this is a viable choice for SPNs as well, and used by PIER and PeerDB. XML as data exchange format for the Web is also favored by several systems, e.g. Piazza and XPeer. As we have seen in section 2.3, RDF is designed for the interconnection of distributed knowledge. Therefore, it is a good match for P2P information systems and has been chosen as data model in several systems, e.g. RDFPeers, SQPeers, and NeuroGrid [79]. Piazza can also support RDF data sources [68].

Another level of expressiveness and complexity is added by the use of ontologies as a schema language that allows the derivation of implicit information by means of reasoning. Ontologies are encoded using concept-based formalisms that support some form of inheritance reasoning. In particular, the use of ontologies as a schema language for describing information is gaining importance [22, 161, 56]. The expressiveness of the respective formalisms ranges from simple classification hierarchies to expressive logical formalisms. The SPNs supporting full-fledged ontologies are DRAGO [22] and coDB [56].

3.2.2 Query Language

The expressiveness of the query language supported by the system is a crucial characteristic of P2P information sharing systems. A ‘language’ providing only key-based access obviously doesn’t make sense in a SPN. However, support for keyword queries enables users to pose queries without requiring schema knowledge, and thus is useful in SPNs, too. And indeed do digital library online services often provide keyword-based search and try to find matches to the keywords in all available attributes (e.g. [41, 89, 128]).

[3] In this view, systems supporting only one-dimensional search, e.g., DHTs, can be seen as a special fixed schema case, where the schema consists of a single attribute

Of course, the choice of query language depends on the data model: relational systems often use a subset of SQL, and RDF based approaches one of the Semantic Web graph pattern matching languages described in 2.3.2. These languages allow to formulate complex multi-attribute queries similar to the ones usual in traditional database systems. Only few of the current SPNs provide a proper high-level query language interface; in many cases, the implementation is just a proof-of-concept, and queries are hard-coded in a proprietary fashion.

A further increase of expressiveness is provided by systems that support full-fledged logical queries, enriched by user-specific axioms, e.g., deductive Semantic Web languages (cf. 2.3.2). This allows the user to explicitly state background knowledge and to introduce new terminology when querying the system. This ability can be used to automatically enrich user queries with information from a user profile to support personalization [50].

Another dimension on which query capabilities can be increased is the relevance dimension. In information retrieval, ranking hits according to relevance metrics is common, and also in databases work in this area has started, e.g., top-k, skyline and preference queries. See chapter 6 for more information.

3.3 Data Integration

In a distributed system it often cannot be guaranteed that the information provided by different sources is represented in the same way. This leads to the need of providing integration mechanisms able of transforming queries and/or data from one representation to another. This problem has been extensively researched in the context of federated databases (cf. 2.2). The basic idea is to define mappings which can be used to translate (re-write) between the different schemas. Although (Semi-) automatic mechanisms, such as lexical analysis have been proposed for SPNs [129], the usually assumption is that mappings are manually created..

The typical approach in SPNs for this translation is query re-writing. Instead of transforming the data to be queried, the query expressions received from external sources are transformed into the format used by the queried source using the mappings between the schemas [68, 3, 56]. This approach still requires a transformation of data in order to make the result of the query compatible with the format of the querying sources.

Most SPNs supporting data integration use simple equality or subsumption statements between schema elements. Approaches that use more complex mappings (in particular conjunctive queries [68]) do not scale to a large number of sources. Serafini et al. have proposed a very expressive mapping formalism, the Local Relational Model (LRM, [162]), which relies on a first-order-logic formalization of the relational model.

3.4 Overview of Schema-Based P2P Algorithms and Systems

In the next subsections, we will describe relevant related systems with specific focus on their respective design decisions. The list of systems is not complete; several further SPN approaches [56, 71, 182, 14] are been described here in detail, because only preliminary results have been published yet.

3.4.1 Systems Focused on Query Processing

PIER The vision for PIER [75, 74] is to provide a distributed relational query processor, based on P2P techniques. In PIER, DHTs are used to index all tuples stored in the system. They were first organized as in CAN [141], now the Pastry derivate Bamboo [144] is used. Tuples are forwarded to the node responsible for its respective key interval. Therefore, data does not stay at the providing node, but is distributed according to the DHT algorithm. For each indexed attribute, a separate DHT namespace is used. To compute a join, a temporary DHT namespace is created, and all relevant tuples are rehashed based on a concatenation of all join attributes. Depending on selectivity of the join subqueries, that operation may involve a substantial fraction of all peers in the network. PIER is efficient regarding the evaluation of selections with equality predicates, but doesn't support range queries. Aggregate computation is supported via hierarchical aggregation. For this purpose, a network spanning tree is maintained together with the DHT structure, as described in [33]. The aggregation query is broadcast via this tree, and parents aggregate results of their children, until the final result is aggregated at the root node. Currently PIER holds all tuples in memory, which limits the storage capacity of the network. On the other hand, introduction of a buffer manager for page management on disk wouldn't require a modification of the basic query processing algorithm. PIER does not have any schema catalog, therefore query plans have to be assembled programmatically; a parser supporting an SQL subset could be added, but would require the provision of catalogs at each peer, and thus increase maintenance overhead.

RDFPeers/MAAN RDFPeers [30] exploits a variant of Chord to facilitate multi-attribute queries called MAAN (Multi-Attribute Addressable Network, [31]). A query is supposed to consist of several subqueries, one for each attribute involved. The most selective subquery is used for initial query routing within the DHT, then the matching tuples are forwarded to the node(s) responsible for the next selection criteria, and so on, until the intersection between all subquery-selections has been computed. For m attributes with associated subquery selectivities s_i this takes $O(\sum_{i=1}^{m}(\log n + n \cdot s_i))$ steps. If these selectivities are low, the total effort may exceed broadcasting costs.

In contrast to Chord, MAAN uses locality-preserving hash functions. To resolve a range query, the query is sent to the node responsible for the lower bound, and then forwarded along the ring to the next neighbor until the upper bound is reached. Each node on the way sends matching tuples to the query originator. Skewed value and query distributions may lead to uneven node load distribution. As [19] have shown, this can't be mitigated by reducing the key space intervals for frequent keys, because that would increase the hop count exactly for the frequent queries, and range queries in that area would have to be distributed to more nodes.

To store RDF graphs within such an infrastructure, a simple schema consisting of the three attributes subject, predicate and object is introduced, and all statements are stored in this table according to the MAAN algorithm. RDFPeers can evaluate subgraph pattern matching algorithms, as specified in RDQL. However, a query parser is not implemented. In the case of RDF, the load balancing problem is of high relevance, because some URIs, e.g. rdf:type or dc:title, occur very frequently. In RDFPeers, a popularity threshold is specified and all items occurring more often than denoted by the threshold are just not indexed anymore, an unsatisfactory solution. RDFPeers has only been evaluated in a limited setting (100 peers, 142,772 triples).

Mercury Another topology supporting multi-attribute (range) queries is Mercury [19]. Mercury has a simple data model, where each data item is just a tuple of attribute values (like one table in a relational model). For each attribute, an overlay network is constructed according to the Symphony algorithm [111]. The only difference is that the data is not hashed, but nodes are assigned value intervals. Each data tuple is replicated into each hub and inserted according to its value in the respective attribute. To resolve a query, it suffices to send it to one of the hubs responsible for a query attribute, where it is routed to the node(s) responsible for the query's requested value (range). To facilitate load balancing, the intervals nodes are assigned can vary in size. Mercury creates approximate histograms for frequency estimation based on a sampling approach. These histograms are used to distribute the load evenly between nodes.

SQPeer In SQPeer [85], data always is kept by the owning node. All data is in RDF format, according to (arbitrary many) RDF schemas. Queries are expressed in RQL (RDF Query Language), an SQL-like language for RDF. Each peer publishes a so-called RVL (RDF View Language, [109]), describing the peer's schema fragment. An incoming query is checked against these views and a query plan annotated with data locations is created. Then, this plan is distributed to the relevant peers and the results merged. The authors propose two different topologies for maintaining the view (schema) information. The first option is to use a super-peer topology. In this case, SQPeer can be seen as an extension of our approach described in chapter 4. The only difference is that the queries can be slightly more expressive (taking class

subsumption into account), and query plans can be a bit more involved. The second option is to store the schema information in a DHT. This is similar to [106], and requires interleaved execution of query routing and plan refinement phases, because the required schema information is not completely available at one planning node, but has to be collected. The implementation status of SQPeer is unclear. No evaluation of the approach has been conducted.

PeerDB PeerDB [129] is based on the relational model. Queries can be posed in SQL, and are evaluated locally at each node. Query processing proceeds in two stages: First, to find relevant nodes, a query is flooded to all neighbors. At each node, a keyword matching algorithm is used to match relation and attribute names used in the query against the ones stored at the peer. The best matches are returned to the query originator. The remote relations are now presented to the user in the order of match degree, and the user has to select the most relevant peers. In the second phase the originator sends the query directly to the selected data sources, and gets the results back. Based on query statistics, neighbor connections are adapted over time (short-cut approach). PeerDB has only been evaluated on a network with 32 nodes. Therefore, its scalability is unclear. The schema matching approach seems to be pretty ad-hoc: Semantically identical relations may have unrelated names, and relations with the same name may contain different types of entities. Expecting the user to find out correct matches just by looking at relation and attributes names seems to be unrealistic.

3.4.2 Systems Focused on Schema Mapping

Piazza Piazzas main goal is to map between different schemas in an open distributed environment [68]. Data stays at the owner's database, but schema information, especially mappings between schemas, are shared. Currently, all mappings are replicated to all peers, thus each peer has a global system catalog. The topology of Piazza is not described in their publications. Piazza's main contribution is a query reformulation algorithm which rewrites an incoming query according to the mappings stored in the global catalog. This is done at the peer receiving the query. The rewritten query is sent directly to peers holding part of the requested data. No performance evaluation of the system has been performed. Piazza uses an XML data model and query language, but can be applied to RDF as well [68], however, with limited query expressivity.

GridVine GridVine [3] is the only mapping-oriented system which uses a DHT as underlying topology, in this case P-Grid. It assumes the RDF/S data model, i.e., all data is stored as statement triples. The indexing and retrieval of these statements is conducted in the same fashion as in RDFPeers: each triple is indexed separately on its subject, predicate and object.

Name	Topology	Data Placement	Data Model	Query Lang.	Refs.
PIER	DHT (Bamboo)	Distributed	Relational	–	[75, 74]
RDFPeers	DHT (MAAN/Chord)	Distributed	RDF	–	[30, 31]
Mercury	DHT (Symphony)	Distributed	Tuples	–	[19]
SQPeer	Super-peer or DHT	Owner	RDF	RQL	[85]
PeerDB	Unstructured	Owner	Relational	SQL subset	[129]
Piazza	Unstructured	Owner	XML	XQery subset	[68, 69]
GridVine	DHT (P-Grid)	Distributed	RDF	–	[3]
DRAGO	Unstructured	Owner	Descr. Logics	OWL subset	[22, 161]

Table 3.1: Overview of Schema-Based Peer-to-Peer Systems

The new aspect of GridVine is its capability to include schema mappings into the query evaluation. Any peer can add property translations, represented in an OWL subset. These subsets are also indexed within the P-Grid. Using the Semantic Gossiping approach presented in [2], incoming queries are not only directly evaluated, but also transformed according to available translations. A proof-of-concept implementation is available, and a first experimental evaluation of a network consisting of 60 peers has been conducted.

DRAGO As explained in section 2.3, the next layer on top of RDF(S) is OWL, a language to specify ontologies based on Description Logics. DRAGO [22, 161] allows to interconnect data sources by specifying so-called *bridge rules*, formulated in C-OWL [24], which translate between their respective ontologies. Based on these rules (and the ontology specifications), queries are evaluated using a distributed tableau reasoning algorithm. The algorithm is proven to yield correct and complete results in a static network. Behavior under network churn is not studied. The system has been implemented as extension of the Pellet reasoner [167]. Performance evaluations haven't been conducted.

3.5 Summary

We can identify two trends in SPNs: on the one hand, a homogeneous schema is assumed, and the query planning and processing challenge is tackled. On the other hand, a heterogeneous schema context is assumed, and consequently schema integration becomes the main challenge. In the latter case, highly expressive mapping languages are used to translate between schemas. However, the scalability of these systems is restricted. Some of the systems focused on homogeneous schemas are scaling significantly better. However, it is very difficult to compare the efficiency of these SPNs, first because no standard benchmark exist, and second, because data model and query expressivity vary largely. Table 3.1 gives an overview about the described systems.

Chapter 4

Super-Peer-Based Query Routing

To achieve our goal of interconnecting library systems in a P2P network, we need an efficient routing algorithm for schema-based queries. The algorithm we propose in this chapter is based on a super-peer topology, where peers connect to super-peers that build up the routing backbone for the whole network.

As we have seen in 2.4, such an approach takes into account the different capabilities of peers with regard to bandwidth, availability, processing power etc. The super-peer backbone is formed by powerful and highly available peers, which take care of indexing and routing (see Fig. 4.1). Information sources such as library catalog systems or digital libraries connect to this backbone; on connection, such a source forwards a self-description to its super-peer which adds it to the distributed routing index, coordinating with other super-peers if necessary. An information consumer can pose a query by sending it to any super-peer. The responsibility of the super-peer backbone is efficient routing of these queries (and their answers), including the distribution and execution of query plans based on the routing index. Further capabilities like mediation and transformation of queries and answers could also be implemented in a super-peer, but are not discussed here. Note that super-peer, information source, and information consumer are not different types of peers, but roles in the network. Any network node can play all of these roles at the same time.

The introduction of super-peers in combination with routing indices at these peers reduces the workload of peers significantly by distributing queries only to the appropriate subset of all connected peers [43].

In the following, we will use the shorthand SPQR for our super-peer-based query routing approach. This chapter includes the description of the topology, algorithms and index structures of SPQR, as well as an evaluation of the approach with respect to super-peer load distribution. Some implementation aspects and the application to the digital library context are presented

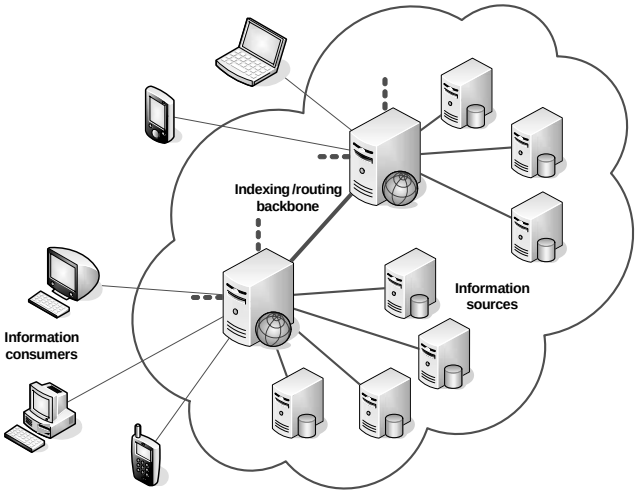


Figure 4.1: Peer roles within a super-peer network

in Chapter 5. The approach presented here restricts queries to metadata elements. An extension which also takes full-text keyword search and result ranking into account is described in Chapter 6.

4.1 Assumptions

As we have argued in Chapter 3, the creation of a full-fledged PDMS which provides similar functionality as a central DBMS is still pretty far away. To devise an efficient approach for distributed schema-based query processing, we need to restrict ourselves to the needs of a specific application context, in our case digital libraries. According to the characteristics and requirements identified in Chapter 1, we base our work on the following assumptions:

- *Comparatively small number of peers* In contrast to popular file-sharing networks, we don't need to build a network which scales to millions of peers; our expectation is that a network of several ten thousands of peers would cover a significant fraction of libraries available via Internet.
- *Schema Homogeneity* We assume that all libraries use the same schema to capture document metadata: nearly all libraries are able to provide a subset of their metadata in Dublin Core format. Obviously this assumption is a simplification: Depending on their topics, libraries can offer more than the typical title/author/date/... metadata set. From a technological point of view, mapping between heterogeneous schemata in a P2P setting is challenging and can be viewed as a still unsolved research problem. although first steps have been taken in that direction, as described in 3.3. Therefore it seems reason-

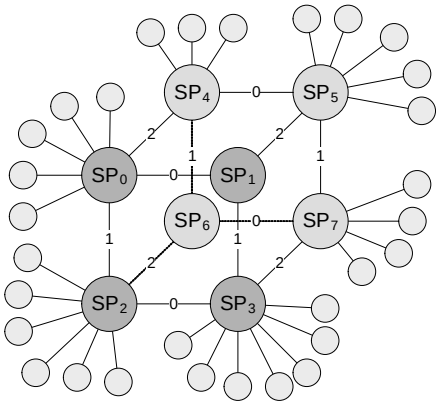


Figure 4.2: Sample HyperCuP Super-Peer Topology

able to start small and simple, and leave aside support for data integration. We allow queries to refer to additional metadata elements provided by libraries, but only as far as they agree on a common schema for them. Usage of other schemas, e.g., for multimedia metadata, is supported as well, but the peer using such a schema exclusively would form an isolated community, although technically connected to the same network.

- *No data movement* Often, documents in an archive are protected by intellectual property rights. Therefore, many libraries are not willing to allow replication of their documents to other peers on which they have no influence. Our P2P infrastructure needs to take this constraint into account; consequently, queries need to be executed directly at relevant libraries, because only they have access to all data required.

- *Selection and projection prevalence* Archive metadata typically is self-contained and does not reference other data sources. Therefore, we support only partial joins directly at the data providers, but no full join across data sources, similar to the constraints used in the design of TSIMMIS [136]. Extended support for distributed joins has been added to our infrastructure by Dhraief et al. [27].

- *Cooperative peers* We rely on the the peers being cooperative. While it is important to take malicious peers into account, this is out of scope of this work.

4.2 The HyperCuP Super-Peer Topology

In any super-peer network, the question is which topology to chose for the super-peer backbone. Here, choosing a DHT topology would not be appropriate, because we want to support complex queries. Therefore, we need a topology suitable for efficient filtered flooding (cf. 2.4.3). In our approach, super-peers are arranged in a hypercube topology, according to

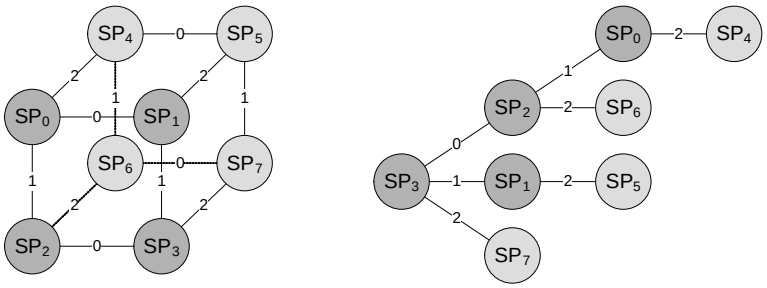


Figure 4.3: HyperCuP and one of its spanning trees

the HyperCuP protocol [158]. HyperCuP is chosen as super-peer topology because it offers an efficient and guaranteed non-redundant query broadcast, which we will then be restricted by the use of routing indices, as described in section 4.4 and 4.5. A small example network is shown in Figure 4.2.

HyperCuP broadcasting works as follows: All edges are tagged according to 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. In this way, for query distribution each node forms the root of a spanning tree which covers the whole super-peer network, as shown in Figure 4.3 for super-peer SP_3.

The topology allows for $O(\log n)$ path length and $O(\log n)$ number of neighbors, where n is the total number of nodes in the network (i.e., the number of super-peers in our case). Moreover, the topology is vertex-symmetric and thus features inherent load balancing among super-peers. Thus, we can use the topology to carry out efficient communication and message forwarding among super-peers. Alternatives to this topology are possible, as long as they guarantee the spanning tree property for the super-peer backbone, which we exploit for maintaining our routing indices and for building our distributed query plans.

A new super-peer is able to join the network by asking any other, already integrated super-peer which then carries out the peer integration protocol. $O(\log n)$ messages are sent in order to integrate the new super-peer and maintain a hypercube-like topology. Any number of super-peers can be accommodated in the network: If some peers are 'missing' in order to construct a complete hypercube topology which consists of 2^d nodes in a d-dimensional binary hypercube, some super-peers in the network occupy more than one position on the hypercube. This means that they have to hold the routing indices for that position, and need to forward messages accordingly. When a new super-peer joins the network, it either fills a gap in the hypercube topology by taking over a position of a node holding several positions, or the dimensionality of the hypercube is extended. [157] describes in detail how the HyperCuP topology is maintained.

4.2.1 Registering Peers at Super-Peers

In order to avoid broadcasting a query to all connected peers of a super-peer we introduce super-peer/peer routing indices that are used to forward the query to relevant peers only. These indices store information about metadata usage at each peer. This includes schema information such as schemas or attributes used, as well as possibly more fine-grained indices on attribute values.

On registration a peer provides the super-peer with its self description by sending a registration request. This request encapsulates a metadata-based description of the peers' properties and content. As the registration may involve quite a large amount of metadata, we build upon the schema-based approaches which have successfully been used in the context of mediator-based information systems (cf. 2.2). The peer's self-descriptions are used to build the super-peer's routing indices.

To cope with network churn, registration messages become invalid after a defined expiration period, and peers have to renew their registration periodically. Without renewal, the peer is removed from the routing indices. By invalidating the peers' registrations periodically we chose a behavior similar to other protocols for dynamic settings (e.g., DHCP) where nodes may leave the network without any notice. If a super-peer fails, its formerly connected peers must re-register at another super-peer (see 4.6.2).

Self-description granularity There is always a trade-off between self-description size and detail. A more fine-grained self-description reduces the probability that queries are forwarded to irrelevant peers, but increases the index maintenance overhead in terms of bandwidth and storage size. An obvious choice for the most coarse-grained description elements are a peer's supported attributes of a schema. Being more coarse-grained doesn't seem to make much sense, because schema information is already very compact. More fine-grained descriptions can be useful for certain types of attributes:

- For attributes on which a total value ordering is defined, a set of value ranges can describe the coverage provided by a peer.
- For enumeration types (e.g., a language attribute), a list of supported values can be stored.
- A special case are attributes referring to taxonomies, a frequent case for the *dc:subject* attribute, but also occurring otherwise, e.g., in educational metadata. To cover this case, a peer can provide a list of base topics. The subsumption relationship is then used to infer if a subtopic is supported by this peer. To compute such inferences, the super-peer needs to store the referenced taxonomy, too.

Granularity	Index Entry Examples
Attribute	dc:title books:translationOf
Attribute Value Range	dc:date[1896,2005]
Attribute Value	dc:language{“de”, “en”}
Attribute Base Value	dc:subject{http://ddc.loc.org/830}[1]

Figure 4.4: Examples for Different Index Granularities

Table 4.4 shows examples for different index granularities. Note that by introducing more fine-grained indices, the information at super-peers moves gradually from a database catalog to full database indices. With few exceptions (e.g., enumeration types with small value domains such as `dc:language`) we do not expect that super-peers provide full table indices, because that would duplicate a significant fraction of each peer's database fragment and thus cause too much effort to store and maintain at the super-peer. However, existing techniques such as histograms [59] or bloom filters [121] could be used to create aggregate/approximate indexes on value granularity level.

In the following, we restrict the formal description of indexing and routing to the schema attribute level; formalization of more fine-grained indices would proceed along the same lines and yield no additional insights.

4.3 Model

The model used here to describe SPQR indexing and query routing is an extension of the relational model described in 2.1 , although our actual implementation uses RDF as data exchange format. This has the following reasons:

- The relational model is simple, very well understood and provides all required abstractions.
- Bibliographic metadata has a rather simple structure, and current standards don't rely on ontology-based schemas. Therefore, our system exploits only the 'Web' aspects of RDF, such as unique identifiers, distributed information, and shared schemas. The 'Semantic' aspect (subsumption relationship or even full-fledged ontology support provided by OWL) isn't used.
- Our algorithm requires no Semantic Web specific features, and could be used equally well to connect relational or XML data sources (the latter as long as the XML data conforms to well-defined XML schemas).

[1] DDC class for 'German Literature'

[69] use the same split between algorithm description and system implementation.

We start with some basic definitions. Let

$P = \{p_1, \ldots, p_n\}$

be the set of all peers (data sources). We can view the data available in the network as one (virtual) database,

$DB = \{R_1, \ldots, R_m\}$,

where R_i is a relation, possibly (and typically) distributed over several peers. Let

$A_i = \{a_{i,1}, \ldots, a_{i,o}\}$

be the set of R_is attributes. Then

$S = \{A_1, \ldots, A_m\}$

is the database schema. Peers hold relation fragments which can be vertically and/or horizontally restricted. We denote the set of R_is attributes provided by peer p $A_{i,p}$. The schema fragment supported by a peer is

$S_p = \{A_{i,p} | A_{i,p} \neq \emptyset\}$.

We use a peer's name as shorthand for the selection function yielding its horizontal fragment. Thus the fragment of R_i stored at peer p is

$R_{i,p} = \sigma_p(\pi_{A_{i,p}}(R_i))$.

Our goal is to process queries over this distributed database in an efficient manner.

4.4 Index Structures

4.4.1 Super-Peer / Peer Routing Indices

The first level index of SPQR, called SP/P index, describes the characteristics of all peers connected to a specific super-peer, and thus guides the forwarding of queries from a super-peer to a connected peer. For complete query routing this summary has to be complemented by information regarding the connected super-peers. We describe these in the next subsection. We use the superscript 'P' to denote information derived from peer summaries.

Let P_k be the set of all peers connected to a super-peer sp_k. For each attribute $a_{i,j}$, the super-peer maintains a set of provider peers

$Prov^P_{a_{i,j},k} = \{p \in P_k | a_{i,j} \in A_{i,p}\}$.

As soon as a peer registers, the super-peer updates its SP/P index, adding the new peer to all $Prov^P$ sets that relate to its supported schema fragment (see 4.6.1).

We define the supported schema of a super-peer as summary of the supported schemas of its peers: As with peers, A^P_{i,sp_k} denotes the set of attributes of R_i provided by super-peer sp_k:

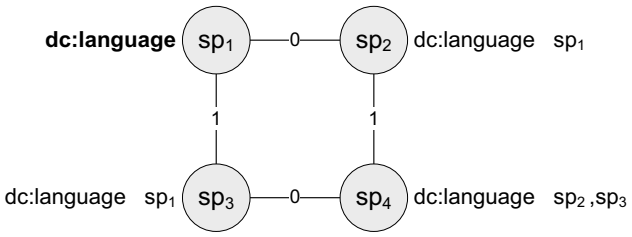


Figure 4.5: Naïvely indexing schema information

$A^P_{i,sp_k} = \left\{ a_{i,j} | Prov^P_{a_{i,j},sp_k} \neq \emptyset \right\}$

The schema fragment supported by a super-peer with respect to its peers is

$S^P_{sp_j} = \left\{ A^P_{i,sp_j} | A^P_{i,sp_j} \neq \emptyset \right\}$.

4.4.2 Super-Peer / Super-Peer Routing Indices

While SP/P indices optimally determine query distribution from super-peers to peers, we need additional indexes to restrict query propagation within the super-peer backbone. They basically contain the same kind of metadata information as the SP/P indices. However, they can't be defined just as summary of the supported schema of neighbor super-peers, because this would lead to unnecessary query forwarding, as we can see from the example in Figure 4.5.

Suppose sp_1 is the only super-peer with peers connected providing the dc:language attribute. The neighbor super-peers sp_2 and sp_3 have to add sp_1 to their respective SP/SP indices as provider for dc:language. Lets assume that we forward that information to sp_4 as shown. Now, if sp_4 needs to distribute a query related to dc:language, it will send it to sp_2 *and* sp_3. But only sp_3 will forward it to sp_1, according to the HyperCuP routing algorithm. Therefore, the message to sp_2 was wasted, and should be avoided.

As the example shows, we need to take the super-peer backbone routing algorithm into account to create optimal SP/SP indices. In the HyperCuP protocol, this means to introduce the edge dimension as additional parameter to the routing index.

Let SP_k be the set of all super-peers connected to sp_k, and d the dimension of the edge where a query comes from; if the query originated directly at sp_k, this is indicated by $d = -1$. A query which came in over dimension d is only forwarded via the edges with dimension $e > d$. We denote this set of neighbor super-peers as

$SP^d_k = \{sp_l | sp_l \in SP_k \wedge \text{edgedim}(sp_l, sp_k) > d\}$.

edgedim(sp_i, sp_k) is the dimension of the edge between sp_i and sp_k according to the Hy-

perCuP protocol. SP_k^d is the set of super-peers to which a query arriving via the edge d is possibly forwarded. Now, the attributes supported by super-peer sp_k with respect to its super-peer neighbors *and* query source, i.e. edge dimension, are:

$A_{i,sp_k}^d = A_{i,sp_k}^P \cup \left\{ A_{i,sp}^{d+1} | sp \in SP_{sp_k}^{d+1} \right\}$.

The definition is recursive, but well-defined: There is a finite number of super-peers in the network, thus the number of hypercube dimensions has to be finite as well, and the recursion can't continue infinitely.

We define SP/SP routing indices as follows:

$Prov_{a_{i,j},sp_k}^d = \left\{ sp \in SP_k^d | \exists e > d : a_{i,j} \in A_{i,sp}^e \right\}$. $Prov_{a_{i,j},sp_k}^d$ contains all super-peers relevant for a query related to $a_{i,j}$ which arrives at sp_k via edge d. The schema summary required as self-description for the super-peer neighbor with respect to dimension d is

$S_{sp_k}^d = S_{sp_k}^P \cup \left\{ A_{i,sp_j}^d | A_{i,sp_j}^d \neq \emptyset \right\}$.

$S_{sp_k}^d$ is the schema fragment supported by sp_k for queries arriving via edge d.

Queries are forwarded to super-peer neighbors based on the SP/SP indices and sent to connected peers based on the SP/P indices.

4.5 Query Routing

In this section, we show how SP/P and SP/SP indices are used to distribute a query. The goal is then to forward a query only to appropriate peers which can answer it. This can be achieved by matching the query elements against the indices and forward the queries only to those super-peers and peers which support the elements contained in the query. A match means that a peer understands and can answer a specific query, but does not guarantee a non-empty answer set. The algorithm described here handles queries which can be evaluated without joining data from several peers. An extension for distributed join support is presented in [27].

The first step in query distribution is the analysis of the query. A set Attr(q), containing all attributes referred to in the query, is created. At each super-peer sp_k receiving this query, it is forwarded to all peers supporting the attributes used. This is the intersection of provider sets for each attribute referenced:

$$Prov_{q,k}^P = \bigcap_{a \in \text{Attr}(q)} Prov_{a,k}^P \tag{4.1}$$

For the query distribution within the super-peer backbone, the routing is slightly more complex, because we take the edge dimension into account. A query originating at super-peer sp_k has to be distributed to all super-peers in

$$Prov^0_{q,k} = \bigcap_{a \in \text{Attr}(q)} Prov^0_{a,k} \tag{4.2}$$

When the query comes in at sp_k via edge dimension d, it has to be forwarded to all super-peers in

$$Prov^{d+1}_{q,k} = \bigcap_{a \in \text{Attr}(q)} Prov^{d+1}_{a,k} \tag{4.3}$$

Now we show that all peers possibly being able to answer the query receive it.

Lemma 4.1. *An atomic query q, referencing only to one attribute a is distributed to all peers supporting this attribute.*

Proof. We need to show that all peers in $Dest = \{p | a \in S_p\}$ receive the query.

Every peer is connected to a super-peer. Therefore, due to definition of $Prov^P_{a,k}$, $\forall p \in Dest\ \exists sp_k :\ p \in Prov^P_{a,k}$. Consequently, if all relevant super-peers get the query, it will be forwarded to the right peers due to forwarding rule 4.1. It remains to be shown that the query reaches all relevant super-peers, i.e., all super-peers sp_k for which $Prov^P_{a,k} \neq \emptyset$.

We know that unmodified HyperCuP broadcasts messages to all connected peers. The route from the message originator – the root of a spanning tree – to any super-peer in the network is unambiguously determined by the HyperCuP forwarding algorithm. Without loss of generality, we number all super-peers on the route as $sp_0, \ldots, sp_n$, where sp_0 is the query originator, and sp_n is a super-peer to which a $p \in Dest$ is connected. Due to the HyperCuP routing algorithm, the edge dimensions between these nodes are $0, \ldots, n-1$.

We need to show that $\forall l = 1, \ldots, n-1 : sp_{l+1} \in Prov^{d_l}_{a,l}$, and do this inductively backwards: $Prov^{n-2}_{a,n-1} = \left\{sp \in SP^{n-2}_{n-1} | \exists e > n-2 :\ a \in S^e_{sp}\right\}$. By definition, for any e, $S^e_{sp_n} \supseteq S^P_{sp_n}$, and we know already that $a \in S^P_{sp_n}$. Also, $sp_n \in SP^{n-2}_{n-1}$, because $n-1 > n-2$. Therefore, $sp_n \in Prov^{n-2}_{a,n-1}$, and $a \in S^{n-2}_{sp_{n-1}}$.

Now it remains to show that for any l, if $sp_{l+1} \in Prov^l_{a,l}$, then $sp_l \in Prov^{l-1}_{a,l-1}$. $Prov^{l-1}_{a,l-1} = \left\{sp \in SP^{l-1}_{l-1} | \exists e > l-1 :\ a \in S^e_{sp}\right\}$. That means, we have to find an e such that the condition is satisfied. We choose $e = l > l-1$, Then we have to show that $a \in S^l_{sp_l}$. This is true, because $sp_{l+1} \in Prov^l_{a,l}$, therefore $a \in S^l_{sp_l}$. □

.

Theorem 4.2. *Any query q which refers to attributes* Attr(q) *is distributed exactly to the peers supporting all attributes* $a \in$ Attr(q).

Proof. The proof proceeds in the same fashion as with Lemma 4.1: We show that all peers in $Dest = \{p|\text{Attr}(q) \supseteq S_p\}$ receive the query. $p \in Dest \Rightarrow \exists sp_k : \ \forall a \in \text{Attr}(q) : p \in Prov^P_{a,sp_k}$ Therefore, $p \in Prov^P_{q,k} = \bigcap_{a \in \text{Attr}(q)} Prov^P_{a,k}$.

As before, we need to show now that the query reaches all relevant super-peers, i.e., all super-peers sp_k for which $Prov^P_{q,k} \neq \emptyset$. Again, we number all super-peers on the HyperCuP route from the query originator to sp_k as $sp_0, \ldots, sp_n$, and we show that $\forall l = 1, \ldots, n-1 : sp_{l+1} \in Prov^l_{q,l}$.

We can infer from Lemma 4.1 that $\forall a \in \text{Attr}(q),\ l = 1, \ldots, n-1 :\ sp_{l+1} \in Prov^l_{ai,j,l}$. It follows directly that $\forall l = 1, \ldots, n-1 :\ sp_{l+1} \in \bigcap_{a \in \text{Attr}(q)} Prov^P_{a,k} = Prov^l_{q,l}$. Therefore, due to rule 4.3, q will be forwarded to sp_n.

It remains to show that for any peer p not supporting all $a \in \text{Attr}(q)$, p_l won't get the query. But this is trivially true: Let p not support an a and be connected to sp_k, then $p \notin Prov^P_{a,sp_k} \Rightarrow p \notin Prov^P_{q,k} = \bigcap_{a \in \text{Attr}(q)} Prov^P_{a,k}$. □

Theorem 4.2 shows that our routing algorithm is correct (i.e., distributes every query to all relevant peers) and optimal (i.e., a peer never receives a query it can't answer).

4.6 Index Updates

The previous sections described the routing indices from a 'snapshot' point of view, not taking network dynamics into account. In this section we show how the indices are maintained when changes in the network occur.

4.6.1 Updating SP/P Routing Indices

An update of the SP/P index of a given super-peer occurs when a peer leaves the super-peer, a new peer registers, or the metadata information of a registered peer changes (e.g., new attributes are added).

In the case of a peer joining the network or re-registering, its respective metadata/schema information are matched against the SP/P entries of the respective super-peer. If the SP/P routing index already contains the peers' metadata only the entry timestamp is updated otherwise the respective metadata with references to the peer are added to the index as described.

The case of a peer leaving the super-peer is a bit more difficult, because any peer might leave unnoticed, e.g., due to a network failure. Therefore, all entries are tagged with an expiration timestamp, and all peers need to refresh their registration regularly. If a peer fails to re-register before its index entry expires, it is assumed to have left the network, and all references to this peer are removed from the SP/P index of the respective super-peer.

The following algorithm formalizes this procedure. Suppose, peer p joins at super-peer sp_k. Then, sp_ks routing indices are updated as follows[2]:

```
foreach A_{i,p} ∈ S_p begin
    foreach a_{i,j} ∈ A_{i,p} begin
        Prov^P_{a_{i,j},k} := Prov^P_{a_{i,j},k} ∪ p
    end
end
```

When a connected peer's self-description changes, it only needs to send the delta to the last state, not the whole information.

4.6.2 Updating SP/SP Routing Indices

To update the SP/SP indices in the backbone, each super-peer sends periodic updates to its neighbors, considering the edge dimension of the respective connection. Suppose sp_l is connected to sp_k via dimension d, then it sends $S^d_{sp_l}$ to sp_k, and sp_k updates its indices as follows:

```
foreach A^d_{i,sp_l} ∈ S^d_{sp_l} begin
    foreach a_{i,j} ∈ A^d_{i,sp_l} begin
        foreach e = 0, ..., d − 1 begin
            Prov^e_{a_{i,j},k} := Prov^e_{a_{i,j},k} ∪ sp_l
        end
    end
end
```

Adding new Super-Peers Adding a new super-peer is a bit more complicated. For a new super-peer, the HyperCuP protocol takes care of identifying new neighbors as discussed in [157]. In this process one of the super-peers becomes responsible for integrating the new super-peer. In most cases the new super-peer will fill a vacant position in the hypercube, which has temporarily been administered by the responsible super-peer. This super-peer, which has been holding additional SP/SP and SP/P indices for the vacant position, transfers them to the new super-peer. If the new super-peer opens a new dimension, it takes over some peers from the old super-peer, and the indices are split accordingly. The neighboring super-peers have to

[2] we do not describe timestamp maintenance here

update their indices as well, by replacing the previously responsible super-peer with the new super-peer on the appropriate dimension. Beyond immediate neighbors, no further update is required.

Removing Super-Peers. The HyperCuP protocol also takes care of a super-peer leaving the backbone. We usually assume that the leaving super-peer coordinates this operation, and specifically asks appropriate super-peer(s) (more than one if the leaving super-peer temporarily fills several positions) to administer its position afterwards. In this process the administering super-peers take over the SP/SP and SP/P indices of the leaving super-peer, and the neighbors of the leaving super-peer as well as of the administering ones have to update their SP/SP indices. Again, no update is required beyond the immediate neighbors. Peers of the leaving super-peer reconnect to the super-peer which administers the vacant position.

In the case of unexpected link failure its neighbors determine the "closest" (regarding smallest hop distance) super-peer. This super-peer then coordinates the administration of the open position with the same procedure as described above. Peers of the failing super-peer have to reconnect at some other super-peer, possibly triggering further index update messages.

4.7 A Simulation Framework for Schema-based Peer-to-Peer Networks

While we have shown that our algorithm is optimal with respect to query distribution to provider peers, we wanted to further investigate characteristics of SPQR, especially the load distribution between super-peers. This has been done via simulation experiments. The experiments and their outcome are described in the next section, here the simulator we designed and implemented for that purpose is described.

An analysis of existing simulators [164] showed that none of them is suitable to simulate a schema-based peer-to-peer-network, since they all concentrate on the traffic or information 'flow' on a much lower network level: the observations are made directly on the transport level or by making abstraction or assumptions on the physical network aspects which is too fine-grained. For our purpose we need a way to describe a specific topology in combination with schema-information, so that we can get results for search and routing in SPNs. Therefore, we built our own simulation framework which is specifically designed to cover the following requirements (partly derived from [52]):

Schema-based resource description We assume that there won't be one fixed schema to describe resources in a P2P network, because our simulator is specifically targeted to SPNs.

For our simulation we only assume that a schema is identifiable and consists of arbitrary many named properties. We don't take into account any relation between properties.

Super-peer based topology While the simulator framework is not tied to a specific topology, we provide specific support for super-peer topologies.

Flexible configuration of distributions Any P2P network simulation run makes assumptions about distribution of bandwidth, latency and other node characteristics. In our case, additionally the distribution of schema-related node properties, e.g., attribute frequencies, needs to be configured. Our simulation framework is designed to allow easy configuration of any of these parameters with a variety of distributions, e.g., uniform, Gaussian and Zipfian distribution.

4.7.1 Design

Our framework is based on discrete event simulation. It includes specific support for schema-based messages and indexing, super-peer networks, and configurable connections suitable for P2P network setups.

Schema-based resource description The main goal of our simulation is to experiment with query routing based on schema information. To represent this information, we use schema elements which can be either complete schemas or single attributes.

Query messages don't contain concrete requests, but only the set of attributes used to formulate the request (denoted as Attr(q) in the previous sections). Information sources 'answer' to these requests on a probabilistic basis, depending on the schema information used by the provider and the information stored in the query message. For example, our model of a query which asks for *(dc:title= "The Power of Metadata"' and dc:date $>$ "1.1.2000")*, is just a set of the used properties *(dc:title, dc:date)*.

For the generation of such queries a configurable distribution is taken into account. We can set as parameters the number of available relations and their frequency distribution, the average number of attributes per relation and the average number of attributes used in a query.

The same applies to peer content. When a peer is created, we do not assign content to them but only the schema this peer is presumed to support for its content. For this assignment, the same relation and attribute distributions are taken into account as for the query generation. Additionally, the average number of relations and properties (and their standard deviation) used by

a peer can be configured. When a query is received by a peer and matches its assigned schema elements, an abstract response is generated with a configurable probability. This approach allows to simulate the routing behavior without needing to generate huge amounts of test data.

The generated queries are distributed over 'originating' peers so that the distribution to the respective super-peers becomes uniform.

Super-Peer-based Topology The simulation framework assumes a super-peer topology. All provider peers have exactly one connection to a super-peer. The super-peers form their own P2P network. If a conventional network is to be simulated, this can be done by instantiating the super-peer backbone only. The super-peer network topology and protocol is pluggable.

Because super-peers are assumed to be highly available, we don't model their up- and down-time, but simulate using a static backbone. This makes it very simple to create different super-peer topologies because it isn't necessary to implement a full connection/disconnection protocol. Instead, a topology class creates all super-peers and the connections between them on simulation startup. Of course, the implementation for the real network has to consider joining and leaving super-peers. But, as super-peer joins or failures will be rare, their influence on the network performance won't be significant.

In contrast, peers will join and leave the network frequently. We model this by a giving each peer a designated lifetime, which is assigned according to a configurable distribution.

Connection Characteristics In contrast to other simulations our approach doesn't rely on a TCP/IP network simulation, but models connections between peers on a higher abstraction level. Any connection has a bandwidth (specified by messages per second) and a delay (in msec). Both properties are assigned to peers and super-peers according to separate configurable distributions.

All connections are bi-directional. Each peer (including super-peers) has an incoming message queue per connection, a processing queue and an outgoing message queue per connection. We can configure the time necessary to process a message and the number of processors available at a peer. Messages between the peers are modeled as discrete events.

4.7.2 Implementation

Our implementation is based on the discrete event simulation framework SSF (Scalable Simulation Framework [42]). The Scalable Simulation Framework is an open source system which implements basic discrete event simulation concepts.

The SSF defines an object oriented interface to utilize and extend the framework. The *Entity* is the central class in SSF. Entities can have processes for event-processing. In our simulator the peers are implemented using entities. An *Event* changes the status of the system or is used for communication between entities. Regarding our simulator when use the events as messages between the peers. *Processes* are used to handle events during the simulation. An entity can have one or more processes. *In- and out-channels* are the communication channel between the entities. An entity can have several in- and out-channels, which are always connected 1:1. This model is easily extended to provide P2P-specific concepts: Entities form the base for peers, Channels are be used to model peer connections, and message occurrences are modeled as events. Thus our framework provides a higher abstraction level, where users can work with P2P concepts directly without the need to consider simulation-specific matters.

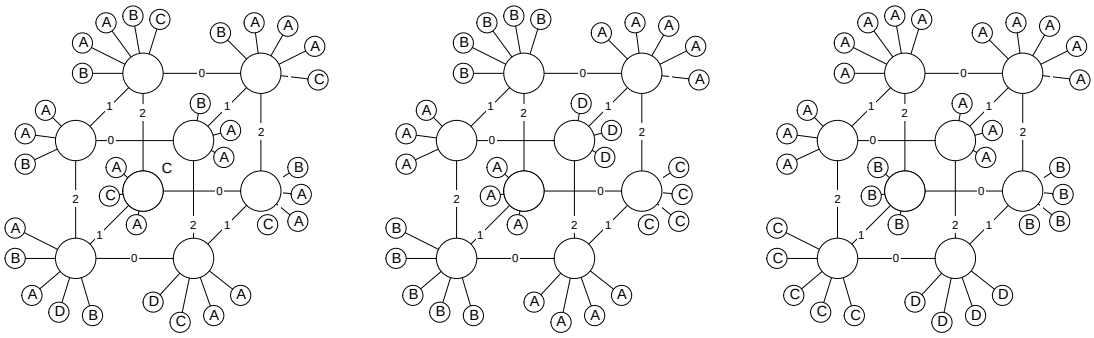
4.8 Evaluation

Query routing in the super-peer backbone depends highly on the distribution of provider peers among the super-peers, according to their supported schema. Intuitively, if peers are distributed randomly, queries have to be sent to more super-peers than if peers are clustered based on their schema. In our experiments, we wanted to measure super-peer routing load and especially the influence of such clustering on their load distribution. We did not consider query processing load, because in SPQR the actual query processing is done by relevant peers, while the super-peers only forward queries and results.

We had the following hypotheses:

1. Clustering peers at super-peers according to their schema will reduce query distribution effort significantly.
2. Clustering super-peers according to their schema (the schema of their peers) will furthermore reduce query distribution effort.

We didn't include a hypothesis about the influence of increasing the number of peers, because in our approach this can already be predicted if peers are clustered at super-peers. In this case, adding new peers does not change the query distribution within the super-peer backbone. As we currently distribute any query to any peer which supports the corresponding attributes, the number of messages between super-peers and peers grows linearly with the number of peers. Therefore, total query distribution effort scales linearly with respect to network size in SPQR.



U *P* *SP/P*

Figure 4.6: Network examples with arbitrary peer distribution (*U)*, peers clustered by schema (*P)*, and peers and super-peers clustered (*SP/P)*

4.8.1 Experiments

We conducted experiments with three different clustering settings, as shown in Fig. 4.6:

- **Peers and super-peers randomly distributed.** In this case peers connect to super-peers in a random fashion, independently of the schema they use. This scenario is abbreviated with U (unclustered).
- **Peers clustered, super-peers randomly distributed**. Here the super-peers collect peers using the same schema. The super-peers are still placed in the hypercube at a random position, regardless of their peers schema information. We try to distribute the load evenly by assigning approximately the same number of peers to each super-peer. Therefore, for rare schemas super-peers will take the responsibility for several schemas. We use the short-hand P (peers clustered) for this scenario.
- **Peers and super-peers clustered**. In this variant we try to find optimal positions for super-peers in the hypercube as well, depending on the schema information. We have to optimize the hypercube for the most frequent schemas; a promising approach is sorting the schemas in hypercube dimensions according to their frequency. Dimension 0 is assigned to the most frequent schema, and therefore a query regarding this schema will be in the right partition of the hypercube after one hop. Queries regarding the second most frequent schema are in the right partition after two hops, etc. We refer to this scenario as SP/P (super-peers and peers clustered).

We simulated a network with 64 super-peers and 10000 peers. The number of schemas in use was set to 32 and a response probability of 5% was assumed. While our algorithm allows the usage of more fine grained indexes, we chose to use a simplified scenario where the peer schemas are disjunctive to get more clear results.

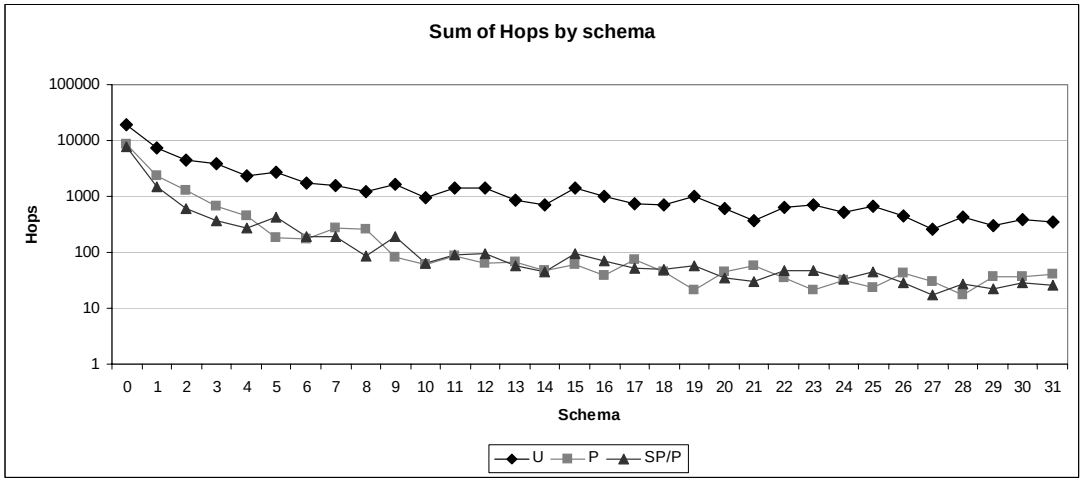


Figure 4.7: Sum of super-peer hops needed to distribute queries

For each scenario 1000 queries were distributed in the network. As the distribution algorithm doesn't depend on previously evaluated queries, the comparatively low number of queries is sufficient to avoid arbitrary results.

The usage probability of the schemas follows a Zipf distribution (skew factor 0.1). Current research [54, 35] has shown that Zipf is the typical distribution in the internet context; this should apply to schema usage as well. The schema distribution is used to calculate the number of peers which use a specific schema to describe their content as well as the number of queries formulated using this schema.

4.8.2 Results

Figure 4.7 shows the sum of hops which were necessary to distribute the queries sorted by schema. For example, to distribute all queries regarding schema 0, we needed nearly 20.000 hops in scenario U, but only about 8500 in P. These results show that clustering peers at the super-peers has a substantial effect on query routing performance. The number of queries a super-peer has to handle on the average is reduced significantly. We can say that hypothesis 1 has been confirmed.

Arranging the super-peers in the hypercube according to their schemas has only a very small effect; hypothesis 2 has therefore not been confirmed. Assigning optimal positions to super-peers in a decentralized and efficient manner seems to be a very complex self-organizing task (especially if the hypercube has to perform a dimension increment or decrement). The first results at least indicate that clustering peers alone is a sufficient optimization.

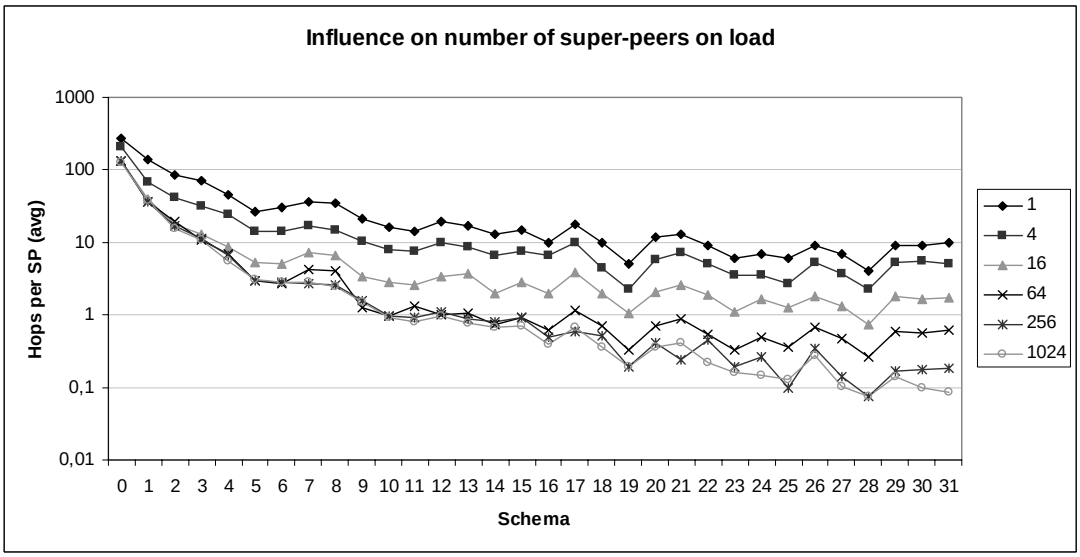


Figure 4.8: Super-peer load for clustered peers (P) in various network sizes

As we saw, clustering reduces the load of the network. However, this comes at a price regarding the load distribution, i.e. the number of queries a super peer has to handle. For scenarios P and SP this load is becomes distributed much more unevenly. The reason is that super-peers responsible for the more frequent schemas bear a higher load, because they get more queries.

As scenario P turned out to be the most interesting, we varied the number of super-peers between 1 and 1024 to evaluate the influence of the backbone size. Fig. 4.8 shows the average load per super-peer for these different sizes. For example, in the case of the 4-node network, on average each super-peer has to process about 200 queries related to schema 0. In the (extreme) case of a 1-node 'network', the super-peer has to process all (273) queries related to schema 0. The average super-peer load is reduced when increasing the backbone size but the gain becomes insignificant for larger networks.

4.8.3 Consequences

In SPQR, queries are sent to all peers possibly able to answer. This results in a linear increase of messages proportional to the increase of the number of peers. As the last experiment shows, we are not able to compensate for this by enlarging the super-peer network.

Therefore, to reduce the amount of processing, we need to restrict the number of responding peers and/or super-peers. We see following options to achieve that goal:

- **Introduction of a *best match* query evaluation approach.** The most promising technique to reduce the network load seems to restrict the number of responses, and only

return a selection of 'best' answers. When sufficiently many good answers are collected, the query doesn't need to be distributed further.

- **Peer preselection** Super-peers could store statistics about the response rate for their peers and forward queries to the most promising peers first. The other peers would get the query only if the first step didn't produce sufficient results.
- **Result Caching** Caching frequent matches and answering from the cache first could also result in a significant improvement. For example, [181] shows that load balancing can be achieved by replicating content within a cluster of peers based on a fairness metric based on content popularity. Other approaches are described in [120] and [169].

In Chapter 6, we present an SPQR extension which combines the first two options to reduce query distribution further.

Chapter 5

A Digital Library Network Prototype for Open Archives

SPQR is a generic algorithm for routing database-like queries in a super-peer-based network. As such, it is just one (albeit a key) building block for a working digital library network infrastructure. A prototype of such an infrastructure, named OAI-P2P, has been designed and implemented, and is described in this chapter.

One of its foundations is the Open Archive Initiative (OAI) standard for digital library access. OAI-P2P employs this standard to allow non-intrusive integration of existing digital libraries into the network. In 5.1 we review the OAI achievements, especially their Protocol for Metadata Harvesting (OAI-PMH), which allows access to a digital library's catalog data. SPQR has first been implemented within the project Edutella, which aims at interconnection of learning object providers within an e-learning network. OAI-P2P is an extension of this infrastructure. We give an overview of Edutella and its query exchange language in 5.2 and 5.3. The OAI-P2P architecture and implementation is presented in 5.4. The chapter concludes with the description of Edutella and OAI-P2P usage experiences in 5.5.

5.1 The Open Archives Initiative Protocol for Metadata Harvesting

Until recently, providing interoperability for digital and print libraries has been limited to the big players; university library systems, scientific publishers and library network cooperatives have the size and the resources to push proprietary protocols or implement large footprint standards like Z39.50 [191]. Smaller institutions do not command these resources.

This is where the Open Archives Initiative (OAI, [91]) stepped in. In order to achieve technical

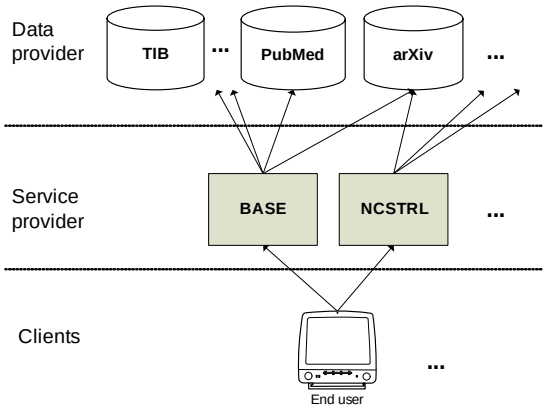


Figure 5.1: OAI-PMH Layered Topology

interoperability among distributed archives OAI has created OAI-PMH which is based on the standard technologies HTTP and XML as well as the Dublin Core metadata scheme [92]. Making use of a number of request types coded in ”verbs” (e.g. ’ListRecords’) the OAI-PMH provides an open interface for metadata exchange and harvesting. In addition to bibliographic schemes like MARC [112] which excel in describing documents in the ”traditional” print paradigm, OAI presently supports the multipurpose resource description standard Dublin Core. As a matter of principle, OAI-PMH is just providing an XML-wrapper for any metadata and can be adapted to both simple and complex and varying metadata sets. One of the main aims of OAI was to keep the protocol simple and easy to implement. The positive feedback and rapid adoption of the OAI-PMH by scientific communities and information professionals have proved this approach right: the number of OAI-enabled digital libraries and library catalog systems is increasing, incorporating smaller libraries as well as areas of science which have not been represented in the earlier attempts.

OAI-PMH is a protocol limited to incremental metadata transfer, providing a technical and organizational framework for metadata harvesting. To keep the instruction set simple, OAI-PMH calls for a separation between data and service providers. Data providers establish an OAI-PMH-based interface to local digital resources, while service providers (like ARC [101] and SCIRUS [160]) provide facilities for searching across multiple archives plus value-added features such as ranking and unified access to other sources. This separation exposes the simplicity of the protocol as the source of its strength (low barrier to adoption) and its weakness: OAI-PMH is designed as simple as possible for data providers at the expense of service providers; creating and maintaining an OAI-PMH service provider requires much more resources than setting up a data provider. On the other hand, OAI-PMH offers no front-end services: data providers offer an interface for metadata harvesting to outsiders but do not have any immediate advantages (like a query service for outside repositories) from their efforts,

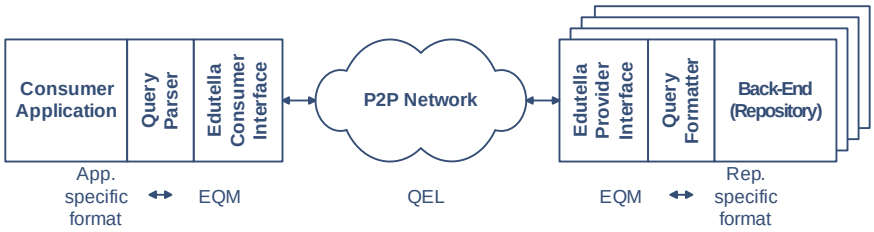


Figure 5.2: Edutella Query Processing Overview

unless a service provider provides an interface to their data.

Figure 5.1 shows a typical OAI topology. Different data providers are harvested by different service providers. All queries are handled by providers at the service provider layer.

When a user wants to query all data providers, he has to send a query to multiple service providers. The results may overlap, and the client will have to handle duplicates. Note also that this architecture makes it difficult for a new data provider to get accessible. As long as no service provider is willing to harvest its metadata, end users won't 'see' them. Another issue occurs when service providers are terminated or reorganized. The most prominent example is Networked Computer Science Technical Reference Library (NCSTRL): the service suffered from limited availability for the best part of 2000 and 2001 (according to [87] due to funding problems). In such a case, the data providers attached to this service provider may find that their archives are no longer harvested, and they lose access to other repositories formerly made accessible by the discontinued service provider. The whole infrastructure has to be re-established with a new service provider.

5.2 Edutella Architecture and Implementation

The SPQR algorithm has been implemented within the open source project Edutella [51]. While Edutella started as P2P network to connect eLearning repositories, it became a generic infrastructure for interconnection of schema-based information providers. The original infrastructure, based on central message hubs [125], has been replaced by an SPQR implementation.

In this network, query and result messages adhere to a specified format, the Query Exchange Language (QEL, see 5.3). However, neither information provider nor information consumer need to support this language directly. Instead, a mediator architecture is employed which transforms queries (and results) between peer-specific format and QEL format. To ease development of such mediators, an object-oriented model of queries, the Edutella Query Model (EQM) has been developed. Consumers and providers only need to convert to rsp. from EQM, all message creation and parsing is done within generic consumer and provider services. Query

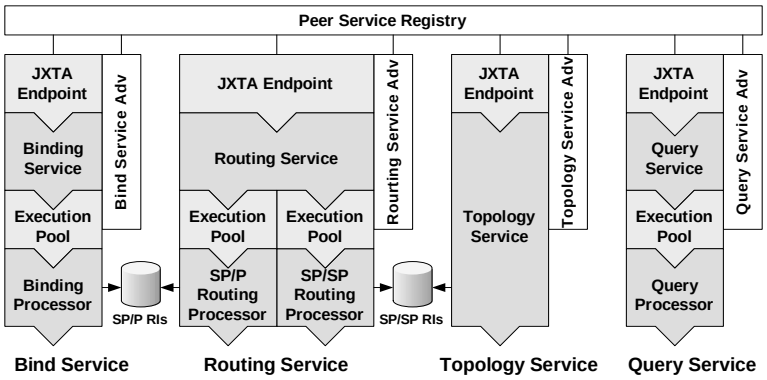


Figure 5.3: Super Peer Service Configuration

distribution is done within the SPQR network. Figure 5.2 gives an overview of this process.

To validate our approach with a prototypical implementation, the Edutella system described in [125] has been redesigned significantly. The monolithic software design has been replaced by a service-oriented architecture. Peer services are plugged together dynamically from partly generic, partly specific components. Figure 5.3 shows a configuration for a super-peer.

Super-peers provide the following services:

- *Bind Service.* The bind service handles peer registration. Provider peers call this service with their self-description to establish the connection to a super-peer. The bind service then updates the SP/P indices according to the SPQR algorithm. It also takes care of initial hand-shaking between peer and super-peer and manages index entry expiration.
- *Routing Service.* This service is responsible for message distribution. It routes queries to the appropriate peers and super-peers, based on the indices created by the binding and topology service.
- *Topology Service.* The topology service takes care of maintaining the super-peer network topology, and also keeps the SP/SP indices up-to-date. If a new super-peer starts, its topology service connects to the topology service of another super-peer, and the location of the new super-peer in the network is negotiated (cf. section 4.2). Afterwards the new neighbors exchange initial SP/SP routing information, and the SP/SP indices are
- *Query Service.* The query service provides a defined interface to issue new query requests within the network. These requests are then distributed via the routing service.

Communication between service components within a peer is done by sending events to monitoring listeners, according to the Observer design pattern. On startup, the components for the configured services are instantiated and the necessary observer relations are created. For example, in the super-peer configuration the query service doesn't have to know how the query

is finally distributed. Instead, the routing service is tagged as *matching actor* for query events in the configuration, and thus the routing service is registered as listener at the query service. This way queries received by the query service are passed to the routing services as query events.

The same query interface is provided by peers, too, but in that case it is configured so that queries are forwarded to data store wrappers which translate and evaluate the query and produce result sets according to the QEL specification.

5.3 A Query Exchange Language

QEL is used within Edutella to express queries against data sources using the Resource Description Framework (RDF). All queries transmitted within the network are represented in this language.

QEL differs from many other query languages for RDF data in that it is derived from datalog [37, 58], a language for expressing database queries based on predicate logic. In this section we provide a short overview of QEL; a comprehensive specification is given in [130].

5.3.1 Queries

Basic RDF Terms There are three kinds of atomic terms in QEL: anonymous RDF resources, non-anonymous RDF resources and RDF Literals. QEL uses N3 syntax for these basic RDF constructs ([15], cf. 2.3.1).

Predicates The most basic construct in datalog are predicate expression. In datalog, predicate expressions consist of a predicate symbol, followed by an argument list, for example

```
name(matthias, "Matthias")
age(matthias, "26")
male(matthias)
```

Arguments can be *values* (e.g. ”Matthias”), *instance names* (e.g matthias) or *variable names* (e.g. X). We use an adapted syntax to cater for the RDF context: Predicate symbols are always identified using a URI reference, Constant names have to written as resource URLs, and the literal syntax described is used for constants. Variables are always starting with an upper-case letter. Thus, the sample predicates above could be written in QEL as

```
@prefix fs:  <http://my.domain.org/path/family-schema#>
@prefix myfamily:  <http://www.myfamily.org/#>

family:name(myfamily:matthias, "Matthias")
family:age(myfamily:matthias, "26"^^<xsd:int>)
family:male(myfamily:matthias)
```

Query Expressions If a predicate expression contains one or more variables as arguments, it is called query literal. Query literals evaluate to true if we can arrive at a fact by replacing the variables with values or constant names, which is a tuple with some of the terms replaced by variables. For example, to ask for all children of Matthias, one would write

```
?- family:father(myfamily:matthias, X)
```

Note that in contrast to Prolog, a datalog query returns all possible variable bindings, not only the first one.

A list of query literals, separated by comma, forms a conjunction (the query evaluates to true if all query literals evaluate to true) called the query body:

```
?- family:name(Person, "Peter"), family:father(Person, Child)
```

This query finds all persons which are named ”Peter” and are fathers. As a side effect, the query returns not only these persons (in the Person variable), but also their children (in the Child variable).

Rules and non-data predicates In datalog, it is possible to introduce predicates that are not part of the data, but derived (sometimes called intensional predicates). This is done by describing rules for their validity. The example shows the rules for being siblings: two persons are siblings if they share either the father or the mother.

```
family:sibling(X, Y) :- family:father(X, Z), family:father(Y, Z)
family:sibling(X, Y) :- family:mother(X, Z), family:mother(Y, Z)
```

Several rules with the same predicate essentially represent disjunctions. Rule definitions are allowed to be recursive; however, lower QEL compliance levels do not support recursive query evaluation (see 5.3.2).

Built-in Predicates QEL contains a number of built-in datalog predicates that can be used to query the RDF data. There are two kinds of built-in predicates, matching and constraint predicates.

Matching predicates are predicates that create new bindings of variables from the RDF data source. The most important matching predicate is *qel:s(X, Y, Z)*, which evaluates to true if the triple (X Y Z) exists in the RDF data.

Constraint predicates only constrain values that have already been found. They cannot find new matches, but act as filter on bindings found via matching predicates. QEL provides predicates such as *qel:equals*, *qel:like* (for string pattern matching), *qel:lessThan*, etc.

Negation Negation in general is not well defined on the Semantic web. The reason is that any data source may add triples breaking the negative assertion. To be well defined, negation requires a closed world, where we know we have access to all data, to be well-defined. Therefore, a general negation operator (while part of datalog) is not part of QEL.

However, we allow negation of atomic constraint predicates. As these do not create more matches but only restrict existing matches, their negation is a well-defined operation even under the open-world assumption. Negation is denoted by ”-” in front of the predicate.

Optional Match In some cases users want to get information if it exists, but don't mandate their existence. Consider a search for resources having a title matching ”physics”:

```
qel:s(X, dc:title, Z), qel:like(Z, "%physics%")
```

The result will only contain the URIs and the corresponding titles. If one would also like to know the subject, he can ask:

```
qel:s(X, dc:title, Z), qel:like(Z, "%physics%"),
  qel:s(X, dc:subject, S)
```

However, not all resources might have a specified subject, and these will not be included in the result set. The problem increases with the number of different extra pieces of information that one might be interested in. Optional match is a solution for this issue: an optional predicate always evaluates to true even if it doesn't match any data, and variables which can't be bound are also set to *null*. In QEL, an optional match literal is marked with a star ”*”. So the above example query can be written as:

```
qel:s(X, dc:title, Z), qel:like(Z, "%physics%"),
  qel:s*(X, dc:subject, S)
```

This query returns all matches to the title constraint, and `dc:subject` values as far as they exist.

5.3.2 QEL Compliance

Not all implementations will be able to support all constructs of this specification. Therefore, QEL compliance levels have been defined which allow to classify information sources according to their query capabilities:

Language	Compliance Level
SQL	Linear Recursive Query (if RDBMS supports SQL3 recursion)
RDQL	Disjunctive Query
SeRQL	Disjunctive Query
RQL	Disjunctive Query
XPath	Conjunctive Query
ConceptBase	General Recursive Query

Table 5.1: Compliance levels of QEL translators

- **Rule-less Query** A rule-less query is a query that does not contain rules.
- **Conjunctive Query** A conjunctive query is a query that contains a maximum of one rule per predicate. It thus does not contain any disjunctions.
- **Disjunctive Query** A disjunctive query is a query that may contains several rules for each predicate, but does not allow for queries to be recursive.
- **Linear Recursive Query** A linear recursive query is a query that contains recursive predicates, but the recursion is linear. This kind of queries is covered by SQL99 [168] expressivity.
- **General Recursive Query** A general recursive query is a recursive query that is not linear recursive. It will require the equivalent of a datalog or prolog processor to be executed.

How implementations have to treat constructs they aren't able to evaluate is specified in [130].

As part of the Edutella project, translators from QEL to SQL, RDQL, SeRQL, RQL, XPath and ConceptBase have been developed. Table 5.1 gives an overview of their respective compliance level.

5.4 OAI-P2P Architecture and Implementation

The main idea behind OAI-P2P is to reuse the advances OAI-PMH has already brought with respect to library systems openness and interface standardization, while mending its weakness with regard to query capabilities and centralized organization of service providers. OAI-P2P uses a mediator approach to achieve these goals. Existing OAI-PMH providers are complemented by a peer application providing the query interface as well as all facilities required to become part of the peer network. In the OAI-PMH view, the peer becomes a small service provider, offering query services for its wrapped OAI data provider. The mediator peer replicates the OAI provider data to an RDF repository and answers queries based on this replica (see

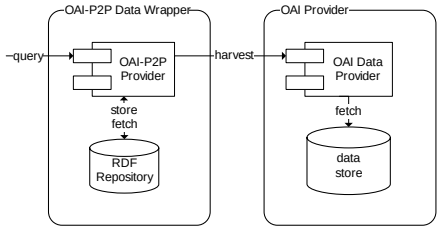


Figure 5.4: OAI-P2P Mediator Peer

Figure 5.4). In contrast to pure OAI-PMH, each data provider sets up its individual OAI-P2P service provider, thus becoming independent from external harvesting services.

The OAI mediator has been implemented as an additional Edutella service which synchronizes an RDF repository with an OAI repository. To increase reuse of this service, it's core is an XSLT-based translator which can be used to convert any XML metadata to RDF. The other part is an OAI-specific protocol adapter; it uses the OAI-PMH request protocol to retrieve Dublin Core metadata sets from an OAI provider. The synchronization service runs as a separate process and checks the assigned OAI provider periodically for updates. If changes have occurred, it updates the RDF repository accordingly. To store and query the metadata, the Jena framework [114] is used which includes a RDBMS-based persistent RDF repository. The translator between RDQL and QEL, which had already been implemented as part of the Edutella project is reused here. Thus, the service- and component-based architecture of Edutella made a high degree of reuse possible in the OAI-P2P mediator implementation.

In the final OAI-P2P system, no hierarchical structure is imposed on OAI data providers. Every provider becomes an equal node in the network, as shown in Figure 5.5.

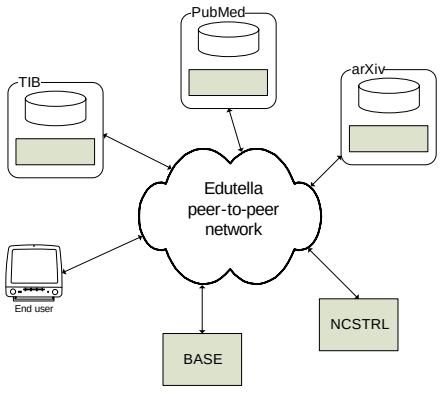


Figure 5.5: OAI-P2P Network

5.5 Experiences

The Edutella and OAI-P2P implementations have been used in the project PADLR [131], an international project with the vision to create a seamless distributed learning environment, where students as well as teachers can manage and publish their personal learning material collections. For this purpose, several tools have been equipped with interfaces to the peer-to-peer network, such as Conzilla [132], a concept-based browser, and SCAM Portfolio [133], a system to create and maintain personal learning portfolios. As testbed, a network consisting of about 20 information providers has been setup, with peers in several locations in Germany as well as in Sweden. The content provided ranges from detailed descriptions of lectures (created within the ULI project) over replicated OAI catalogs, up to direct connections to the Uppsala University Library catalog and to the digital media library (Mediebiblioteket) of the Swedish Educational Broadcasting Company.

From a technical point of view, the system works as expected. Queries are distributed to relevant peers and all matching results are delivered. Early user experience revealed some insights with respects the underlying data and query model. These insights are related to the type of search process: when users pose queries to locate resources they already are aware of, then the network is able to match this information need efficiently. But results for explorative queries were not equally satisfactory. With too loose query constraints, the result set often becomes too large to be of much use for a human user. To arrive at a suitable result set size, the user has to tighten constraints. In this process, it easily happens that the result set gets empty, forcing the user to relax the query again.

This phenomenon is well known for large databases, and has lead to the adoption of the notion of *relevance* from information retrieval in the database area, and to the development of query models capturing such a notion, such as top-k queries, skyline queries, or – as a more general approach – preference queries.

The next chapter shows how the approach presented so far can be extended to accommodate such query models as well, to lead to an improved user experience for explorative queries, and at the same time to more efficient query distribution.

Chapter 6

Preference-based Query Evaluation for Super-Peer Networks

As we have seen in the last chapter, the traditional database model of exact queries, where result sets are limited only based on hard constraints, is not perfectly suitable for exploratory queries, where users are looking for relevant items without exactly knowing what they need. Therefore, query languages like SQL over relational databases have been extended to facilitate rank- and/or score-based retrieval algorithms. These approaches assign a degree of match with respect to user specified soft constraints to each database object and then aggregate the rank/score values to compute the set of best matching answers. Under the exact match paradigm too specific query predicates often lead to empty result sets, while too unspecific hard constraints may yield huge numbers of results. The notion of best matches fits much better to typical user's search requests, because query specificity is in a way automatically adapted to the available content. In this chapter, we will present an extension of SPQR which allows to also evaluate queries with soft-constraints in a distributed setting.

Several ways of extending the usual database query expressivity have been proposed. Top-k queries [53] deliver a well defined set of k best answers according to a user-provided scoring function. They have shown their broad applicability in various areas like Web search engines, mobile database applications, or content-based retrieval in multimedia collections or digital libraries. A formal extension of relational algebra by a specific top-k operator has been proposed in [98].

In some cases of multi-predicate queries, it is difficult or impossible for a user to specify predicate weights for a scoring function. Suppose someone looks for the newest articles about a specific topic. How should the newness be scored in relation to the topic relevance? These cases are an application for skyline queries, where predicates are viewed as independent query dimensions, and best matches are determined according to the principle of Pareto optimal-

ity [29, 177, 135].

The top-k and skyline approaches have been generalized to the notion of preferences-based querying, formalized independently by Kießling [84] and Chomicki [36]. The term *preference* was coined in the context of personalized systems (e.g. [145]), where preferences capture a users likings and dislikes. However, preference-based queries are not only suitable in this context. They also provide a very flexible and expressive way to describe a wide variety of queries for best matches. Therefore, we will use this notion to describe our extension of SPQR to a best match, ranking algorithm[1]

In the following, after introducing a model for preference queries in 6.1, we present the extended SPQR query evaluation algorithm (6.3.1), and its evaluation in a digital library scenario (6.4).

6.1 Preference-based Querying for Relational Databases

To describe the preferences and logical query plans used in extended SPQR, we rely on the preference query formalization proposed by Chomicki in [36]. In this extension to relational algebra, preferences are expressed as binary relations between tuples from the same database relation.

Definition 6.1. Let Let $A = \{a_1, \ldots, a_o\}$ be the set of Rs attributes, and $U_i, 1 \leq i \leq o$ the respective domains of a_i. Then any binary relation $\succ$ which is a subset of $(U_1 \times \ldots \times U_o) \times (U_1 \times \ldots \times U_o)$ is a *preference relation over* R.

We restrict this very general notion to relations that are defined by so-called *intrinsic preference formulas*, first order logic expressions in which a limited set of constraint operators occur.

Definition 6.2. A *preference formula* $C(t_1, t_2)$ is a first order formula defining a preference relation $\succ_C$ such that $t_1 \succ_C t_2 \equiv C(t_1, t_2)$. An *intrinsic preference formula* is a preference formula which uses only the following (built-in) constraint predicates:

- *equality constraints*: $x = y, x \neq y, x = c, y \neq c$, where c is a constant.
- *rational-order constraints*: $x\theta y, x\theta c$, where c is a rational constant, and $x, y \in U_i \subseteq \mathbb{Q}$.

[1]The work presented has been described in a top-k framework in [127, 12]. This framework already had to be stretched to cover composition of keyword and topic soft constraints as proposed in [11]. Therefore, it seemed more appropriate to use the preference model as framework to describe this work here. This does not mean that the proposed approach can be used to evaluate any kind of preference-based query in a distributed context. For example, higher dimensional skyline queries are definitely not in scope of the presented algorithm.

Doc				
Id	*Title*	*Date*	*Topic*	*Body*
LA072689-0030	New China Student Protest	1989-07-26	Politics/ Foreign	In the first known revival of student protests in China...
LA072689-0033	Grant to George Russell well deserved	1989-07-26	Calendar/ Entertainment	George Russell has been the jazz composers'...
LA072689-0041	Wright Case Lawyer may probe Gingrich	1989-07-26	Politics/ National	The House Ethics Committee said it plans to hire...
LA072689-0070	Bray wins Gold on split Decision	1989-07-26	Sports	John Bray knew what he wanted to do...

Figure 6.1: News Archive Schema and Examples

Now we can define the preference query operator, called *winnow operator* by Chomicki:
Definition 6.3. For a relation R with schema A and a preference formula C defining a preference relation $\succ_C$ over A, the *winnow operator* ω_C is defined as

$\omega_C(R) = \{t \in R | \neg \exists t' \in R\ t' \succ_C t\}$.

If more than the strictly best result(s) should be returned, result ranking in a top-k style can be achieved by iterative application of the preference operator:
Definition 6.4. Given a preference relation $\succ_C$, the nth iteration of the winnow operator ω_C in R is defined as:

$\omega_C^1(R) = \omega_C(R)$
$\omega_C^{n+1}(R) = \omega_C\left(R - \bigcup_{1 \leq i \leq n} \omega_C^i(R)\right)$
The union of the k best match sets is:
$Top_C^k = \bigcup_{1 \leq i \leq k} \omega_C^i(R)$

Note that Top_C^k can contain more than k hits, depending on the preference relation C.

6.2 Basic Scoring Functions for Document Search

We use the context of news archives as illustrating example. Each news article has a title, a publication date, a topic such as 'politics' or 'sports', and a body containing the news text. We assume a flat schema here, where each news occupies just one table row of the table Doc (Figure 6.1). On this archive, we can pose queries such as

"Search articles based on the keywords 'London' and 'Olympics', preferring news from the 'Politics' category".

In the following, we will use this example to show how preferences such as in our sample query are expressed formally using the described scoring functions.

Often, preferences with respect to documents can be easily expressed using standard comparison operators. For example, to express the preference for more recent documents, a compari-

son of publication dates can be used:

$d_1 \succ_{recency} d_2 \equiv \text{Date}(d_1) > \text{Date}(d_2)$

However, simple comparison functions alone are not sufficient for document search. We need other atomic scoring functions to express common document preferences. Therefore, we define two built-in scoring functions which allow to express popular search criteria, a relevance function computing a document score with respect to given keywords, and a topic similarity function computing a score with respect to a given topic.

6.2.1 Topic-distance in Taxonomies

In digital libraries and library catalogs, it is usual to classify documents according to their topics, based on a topic taxonomy (a tree structure putting topics into sub-/super-topic relations). This information is highly valuable to identify relevant documents as well. Of course, we can use a topic as query constraint for limiting results to exactly the desired topic. However, we often are willing to relax this constraint if no results show up on the exact topic (taking other constraints into account). This is also a perfect application for the expression of a preference: Documents with a specific topic are preferred over documents related to a similar topic, but we are content with the latter should the former not be found.

Several methods to compute topic similarity have been proposed [78, 99, 100, 143]. We use the formula presented in [99], because it has been evaluated favorably in comparison with human expert judgments.

$$topic\_sim(c_1,\ c_2) = \begin{cases} e^{-\alpha \cdot l(c_1,\, c_2)} \cdot \frac{e^{\beta \cdot h(c_1,\, c_2)} - e^{-\beta \cdot h(c_1,\, c_2)}}{e^{\beta \cdot h(c_1,\, c_2)} + e^{-\beta \cdot h(c_1,\, c_2)}} & : \text{ if } c_1 \neq c_2 \\ 1 & : \text{ otherwise} \end{cases} \tag{6.1}$$

where $l(c_1,\ c_2)$ is the shortest path between the topics c_1 and c_2 in the taxonomy tree and $h(c_1,\ c_2)$ is the depth level of the direct common subsumer. α and β are parameters to optimize the similarity measurement (according to[99], the best setting is usually $\alpha = 0.2$ and $\beta = 0.6$).

Now we can express our preference from the sample query regarding document topics:

$d_1 \succ_{topic} d_2 \equiv topic\_sim\,(Politics, \text{Topic}(d_1)) < topic\_sim\,(Politics, \text{Topic}(d_2))$

6.2.2 TFxIDF as keyword scoring function

The most prevalent search criterion nowadays is keyword search over document full text. This has been supported for decades in information retrieval. One of the basic relevance notions

in IR is TFxIDF (see e.g. [188]). TFxIDF stands for Term Frequency and Inverse Document Frequency and is a content-based ranking method. It calculates the relevance of a document, based on how often a search term appears in a document (term frequency TF), and how often the term exists in the whole document collection (inverse document frequency IDF). The more search terms are found in a document, the more important the document is, taking into account how often the search term is found in the collection, i.e. weighting rare terms in documents higher. A detailed introduction can be found in [188]. For a term t_i from a set of keywords and a document d_j from a document collection T_r TFxIDF is defined as

$$TFxIDF(t,\ d) = \underbrace{n(t,\ d)}_{TF} \cdot \underbrace{\log \frac{|T_r|}{n(t)}}_{IDF} \tag{6.2}$$

where $n(t,\ d)$ denotes the number of occurrences of the term t in the document d and $n(t)$ is the number of documents that contain the term t.

TFxIDF computation requires collection-wide information; as we don't work directly on a document collection, but on a database relation, we need to specify to which collection the scoring should refer. This is done with an additional query expression col. For a given collection specification col we then can define

$$tfidf(t, d, col) = n(t,\ d) \cdot \log \frac{|col|}{n(t,\ col)} \tag{6.3}$$

Using this scoring function, we can express which full-text we prefer based on the keyword condition of our sample query:

$$\begin{aligned} &d_1 \succ_{keywords} d_2 \equiv \\ &\quad tfidf('London', \text{Body}(d_1)), \pi_{\text{Body}}(\text{Doc})) + tfidf('Olympics'', \text{Body}(d_1)), \pi_{\text{Body}}(\text{Doc})) \\ &\quad > tfidf('London', \text{Body}(d_2)), \pi_{\text{Body}}(\text{Doc})) + tfidf('Olympics'', \text{Body}(d_2)), \pi_{\text{Body}}(\text{Doc})) \end{aligned}$$

This expression specifies all document bodies in relation 'Doc' as basic collection.

Due to TFxIDF being always related to a document collection, it doesn't suffice in a distributed setting to compute the score locally at each peer, because that would lead to distorted scores when merging results. Therefore all peers have to be provided with the required collection-wide information. [179] describes in detail how this is done efficiently in our super-peer-based setting.

Formal notion of example query With the defined scoring functions, our sample preference formula looks as follows:

$$d_1 \succ_{sample\_query} d_2 \equiv d_1 \succ_{topic} d_2 \vee (\text{Topic}(d_1) = \text{Topic}(d_2) \wedge d_1 \succ_{keywords} d_2)$$

The query $Top^k_{sample\_query}(\text{Doc})$ will return (about) k preferred documents for the sample search. We will use this preference as query pattern in our evaluation, too (cf. 6.4).

6.3 Progressive, Preference-based SPQR

In this section we will present our algorithm for basic preference-based querying capabilities in the HyperCuP network. According to the distributed nature of the retrieval and the P2P network the distributed retrieval algorithm is divided into three parts that are respectively executed by

- the super-peer initially receiving the query,
- the super-peers in the HyperCuP backbone,
- and the local peers at each super-peer.

Since a dissemination of global knowledge should be avoided due to the overhead of data transmission, a basic concept of our algorithm is to locally evaluate as many parts of the query as possible. This means only the super-peer receiving the query (i.e. the root node of our implicit HyperCuP spanning tree) needs full information to control the execution of the queries in order to guarantee a correct result set with a minimum transmission of data. This super-peer hands on the query to the relevant super-peers along the backbone of adjacent super-peers, which in turn forward the query to their relevant adjacent super-peers and connected local peers, without having to have full information about how the query answering is progressing. The local peers just execute the query over their local database fragment and retrieve some best matching objects. We will present all relevant steps in detail in the following.

Assumptions In addition to 4.1, the algorithm relies on the following assumptions:

- **Uniform Ranking** We assume that every peer throughout the network uses the same scoring functions to compute document scores with respect to a query; input data to compute these scores may be different.
- **Zipfian Query Distribution** Distribution of item popularity, and accordingly, of query frequency on the internet is not uniform, but usually follow a Zipf distribution, where few queries make up the majority of all requests. Zipf distributions are ubiquitous in

content networks, the Internet and other collections and have become one of the most empirically validated laws in the domain of linguistic quantities and networks (in the form of power law distributions) [54, 35, 116, 5].
In P2P networks typical consumers are interested in only subsets of all available content and content categories [44]. Documents are also distributed following Zipf's law, i.e. many consumers are interested in a few most popular resources, while the interest in the large rest of resources is comparatively low. This makes it safe to assume that the occurrence frequency of queries (rsp. the subexpressions they contain) will also follow a Zipf distribution.

6.3.1 Basic Algorithm

Overview A basic question is if distributed computing of a preference query yields the correct result set. To show that this is indeed the case, we start with a simplified version of the algorithm which consists of three steps:

- The super-peer backbone distributes the query to all connected peers.
- Each peer identifies preferred items according to given preference query and sends them to super-peer.
- Super-peers merge and forward the results.

For presentation purposes, we assume that a query consists just of a preference $\succ_C$ on a relation R. Of course, any such query can also include arbitrary hard constraints which act as filter before the preference is taken into account.

Query Distribution Each super-peer uses a simple broadcast to distribute the query. The indices introduced for SPQR can also be used here to restrict query distribution. For example, a preference query only needs to be sent to peers supporting all referred schema elements. However, the preference semantics are not taken into account for initial query distribution.

Local Query Evaluation Any peer p computes the result set for its local database fragment, i.e., $\omega_C(\sigma_p(R))$, and returns this set to its super-peer.

Result Merging Each super-peer collects the responses from all peers and super-peers to which it had distributed the query. As soon as the preferred items have been delivered by the respective peers and super-peers, the super-peer again applies ω_C to the union of all results.

Formally, the following operation is executed on super-peer sp_i, assuming the query came in via dimension d: $\omega_C(\bigcup_{p\in P_i}\omega_C(\sigma_p(R)) \cup \bigcup_{sp\in SP_i^d}\omega_C(\sigma_{sp}(R)))$ [2].

Similar to the non-preference query case (cf. 4.5), we have to show that computing ω_C independently at each peer, and merging results at the super-peers leads to the same result set as if we had computed ω_C on the whole relation.

First, we show a Lemma about constructing a preference result from preference results on relation fragments:

Lemma 6.5. *For any horizontal fragmentation of relation R, denoted as $\sigma_{F_1}(R), \ldots, \sigma_{F_n}(R)$, the following holds:*
$\omega_C(\bigcup_{1\leq i\leq n}\omega_C(\sigma_F i(R))) = \omega_C(R)$.

In other words, the computation a preference relation ω_C by

(1) independent computation of ω_C on the fragments,

(2) union of the resulting answers, and

(3) computation of ω_C on this union,

yields the same result as computation on the whole relation.

Proof. First, we show that $\omega_C(\omega_C(R)) = \omega_C(R)$. Trivially, for any set R, $\omega_C(R) \subseteq R$. Therefore, also $\omega_C(\omega_C(R)) \subseteq \omega_C(R)$. So what is left to show is that also $\omega_C(\omega_C(R)) \supseteq \omega_C(R)$. We prove this by contradiction. Suppose $\exists t \in \omega_C(R) : t \notin \omega_C(\omega_C(R))$.
$t \notin \omega_C(\omega_C(R)) \Rightarrow \exists t' \in \omega_C(R) : t' \succ_C t$. But, $t' \in \omega_C(R) \Rightarrow t' \in R$, and due to $t' \succ_C t$ it follows that $t \notin \omega_C(R)$ ↯

With this result, we can prove that $\omega_C\left(\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R))\right) \supseteq \omega_C(R)$:

$t \in \omega_C(R) \Rightarrow \neg\exists t' \in R\ t' \succ_C t \Rightarrow \forall i\ \neg\exists t' \in \sigma_{F_i}(R)\ t' \succ_C t$ (because $\sigma_{F_i}(R) \subseteq R$). Therefore, $\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R)) \supseteq \omega_C(R)$.

Now, $\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R)) \supseteq \omega_C(R)$
$\Rightarrow \omega_C\left(\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R))\right) \supseteq \omega_C(\omega_C(R))$
$\Rightarrow \omega_C\left(\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R))\right) \supseteq \omega_C(R)$

It is left to show that $\omega_C\left(\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R))\right) \subseteq \omega_C(R)$.
Suppose $\exists t \in \omega_C(R) : t \notin \omega_C\left(\bigcup_{1\leq i\leq n}\omega_C(\sigma_{F_i}(R))\right)$.
$\Rightarrow \exists j : \left(t \in \sigma_{F_j}(R) \wedge \neg\exists t' \in R : t' \succ_C t\right)$
$\Rightarrow \exists j : \left(t \in \sigma_{F_j}(R) \wedge \neg\exists t' \in \sigma_{F_j}(R) : t' \succ_C t\right)$
$\Rightarrow \exists j : t \in \omega_C(\sigma_{F_j}(R))$

[2] notation based on definitions introduced in 4.3.

$\Rightarrow t \in \bigcup_{1 \leq i \leq n} \omega_C(\sigma_{F_i}(R))$.

Now, due our assumption of t being in $\omega_C(R)$, $\neg\exists t' \in R : t' \succ_C t$
$\Rightarrow \forall i, 1 \leq i \leq n : \neg\exists t' \in \sigma_{F_i}(R) : t' \succ_C t \Rightarrow t \in \omega_C\left(\bigcup_{1 \leq i \leq n} \omega_C(\sigma_{F_i}(R))\right)$ ↯

Thus, $\omega_C(\bigcup_{1 \leq i \leq n} \omega_C(\sigma_{F_i}(R))) = \omega_C(R)$. □

Note that $\bigcup_{1 \leq i \leq n} \omega_C(\sigma_{F_i}(R)) \subseteq \omega_C(R)$ does not hold in general, because tuples which are not locally dominated, but would be dominated by tuples stored at other peers, are not removed from the union.

Theorem 6.6. *For a given preference query based on $\succ_C$on relation R, originating at sp_i, and a super peer network as defined in 4.4, the following holds:*
$\omega_C(R) = \omega_C(\bigcup_{p \in P_i} \omega_C(\sigma_p(R)) \cup \bigcup_{sp \in SP_i^0} \omega_C(\sigma_{sp}(R)))$

Proof. Follows directly from Lemma 6.5. □

This result shows that the algorithm described above, based on SPQR, does indeed deliver the result a preference-based query.

6.3.2 Optimized Routing and Merging

In this section, we describe how to efficiently distribute preference-based queries in the SPQR network. Two factors allow to optimize the query processing algorithm further:

- With preference queries, result set sizes are typically reduced significantly, and only a small subset of peers actually contributes to the final result. Therefore, we can reduce distribution effort if we manage to forward queries only to peers holding the best matches, which are going to become part of the result set, instead of forwarding it to all peers supporting the corresponding schema elements (as before).
- when collecting results, often peers can be excluded from further consideration (pruned) already after having provided their single best match, because even this one may be already dominated by a result from another peer.

The first optimization opportunity is used by maintaining an additional *preferred peer index*, based on the results of previous query occurrences . To avoid unnecessary result transmission, we introduce *progressive result fetching*, where additional results are only transmitted on demand.

Preferred Peer Index The idea behind the preferred peer index is that we could avoid a large fraction of distribution efforts if we knew which peers hold the best matches. This is impossible for every query, but it is easy to estimate for repeated queries. In case the network hasn't changed too much a good approximation is to distribute the query to the same peers which had the best answers when this query was evaluated last time. To accomplish this optimization, each super-peer maintains statistics about recent query evaluations. For each query, an index entry is created (or updated) that holds the information which of its local peers and adjacent super-peers contributed results. These index entries can be maintained efficiently even in rather volatile P2P networks to hold sufficiently current information about object distributions, as will be shown in section 6.4. To adapt to changes in the P2P network, the index lets all entries expire after a specified time span. The more volatile the network is the shorter the expiration period has to be in order to adapt to changing data allocations. As in the basic SPQR algorithm, a super-peer needs to take into account on which edge dimension the query came in when forwarding it to adjacent super-peers. Therefore, the super-peer maintains separate sub-indices for its peers (usable for queries from any direction), and for each hypercube dimension. Fortunately, preferred peer index maintenance comes at a negligible price. No additional messaging is required, because the collected information passes through the super-peers anyway. The only cost involved is the memory consumption of index entries, which is rather small because it doesn't consist of the actual results (which may be arbitrarily large) but just peer ids. Therefore, while this index 'only' improves performance for repeated queries, it is in no case harmful.

With the preferred peer index, the algorithm basically looks as follows:

- Super-peers forward query *according to their preferred peer indices*.
- Each peer identifies preferred items according to given preference query and sends them to super-peer.
- Super-peers merge and forward results, *and update their indices.*

Additionally to using the whole query as key to the preferred peer index, it is possible to store preferred peer information about subqueries in separate indexes. This information can be used to compute an optimized destination set for queries which had not yet occurred, too. While this destination set might not be optimal, it can still reduce distribution costs considerably. See section 6.4 for an example.

Progressive Result Fetching Progressive result fetching comes into play if more than the top results are requested. In this case, ω_C has to be executed iteratively on the remaining objects, as described in 6.1. It is obviously not optimal if each peer p delivers the whole set

$\omega_C^i(R_p)$ for each iteration i, because already after the first response it may become clear for a lot of peers that they can't deliver better matches than the one's already seen at their super-peer. As soon as this has become clear, these peers (rsp. their database fragments) should be pruned.

A simple ω_C^2 examples illustrates this: In order to get the dominating object(s) $\omega_C^1(R)$ in the super-peer we only need the dominating objects of all its local peers and super-peers along the spanning tree starting at the query originator. Having chosen the maximum object from any of the peers, we can compute $\omega_C^2(R)$ of this set and exclude all peers which didn't get their $\omega_C^1(R)$ objects placed there. All the other peers still may be able to offer more 'second-best' objects their top-scored. So for determining the objects in $\omega_C^2(R)$, we only additionally need to ask the peers that already contributed to $\omega_C^1(R)$ for their $\omega_C^2(R)$ sets. For higher numbers of query results this process can be repeated inductively until all $\omega_C^k(R)$ sets are delivered, as we show below.

Since the best objects are determined iteratively, the merging super-peer can immediately deliver each batch of result objects to the super-peer directly up the super-peer backbone, enabling it in turn to also return its merged results at the earliest point in time. This successive query result delivery behavior not only optimizes bandwidth use, but also helps to improve the psychologically felt response time for the user by offering correct result objects for consideration already at an early stage.

6.4 Evaluation

While preference queries are more expressive than the ordinary relational calculus, they can also be difficult to compute efficiently, even in a non-distributed setting. Of course, a better result than in a centralized context cannot be expected, therefore it makes sense to chose a scenario for our evaluation for which local computation is efficient.

In our case, we chose a typical digital library context, where users search for documents in a newspaper article collection. Queries are composed from a topic and a keyword preference, as described in the example from 6.2. We used the TREC document collection volume 5 consisting of LA Times articles for our experiments. The articles are already categorized according to the section they appeared in, and we use this information as base for our document classification. To simulate a network of document providers, the articles are distributed among the peers in the network. The simulated network consists of 2000 peers, each providing articles from three categories on average (with a standard deviation of 2.0).

The simulation is based on the framework described in 4.7. TFxIDF calculation was imple-

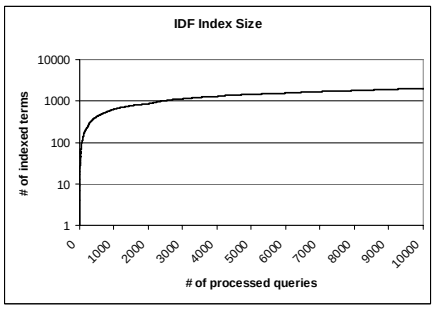


Fig. 1. Index size

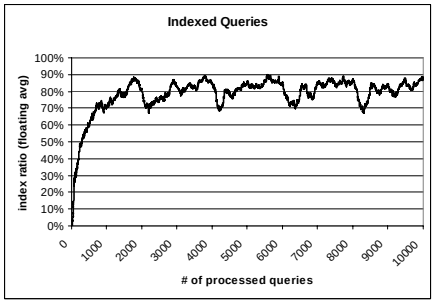


Fig. 2. Coverage of query index

mented using the (slightly modified) search engine Jakarta Lucene [107].

We assume a Zipf-distribution for query frequencies with skew of 0.0. News articles are popular only for a short time period, and the request frequency changes correspondingly. With respect to the Zipf-distribution this means that the query rank decreases over time. Query terms were selected randomly from the underlying documents. In our simulation, we generate 200 new most popular queries every 2000 queries which supersede the current ones and adjust query frequencies accordingly. This shift may be unrealistically high, but serves well to analyze how our algorithm reacts to such popularity changes.

6.4.1 Results

Index size Figure 1 shows how the IDF index at each super-peer grows over time. After 10000 queries it has grown to a size of 2015, only a small fraction of all terms occurring in the document collection. A global inverted index we would have had contained 148867 terms. This underlines that much effort can be saved when only indexing terms which are actually appearing in queries.

Index effectivity Both category and query index become quite effective. After nearly 2000 queries, the query index achieves a coverage of 80%. Figure 2 shows how each popularity shift causes a coverage reduction from which the query index recovers after about 1000 queries. This shows that a change in query popularity over time is coped with after a very short while.

As there are only about 120 different categories, after less then 1000 queries the index contains nearly all of them (Figure 3). We assume that news provider specialized on some topics change these topics only very infrequently. Therefore, peers do not shift their topics during the simulation. Thus, the category index serves to reduce the number of contacted peers continuously, also after popularity shifts.

Figure 4 shows how many peers had to be contacted to compute the result. The influence of

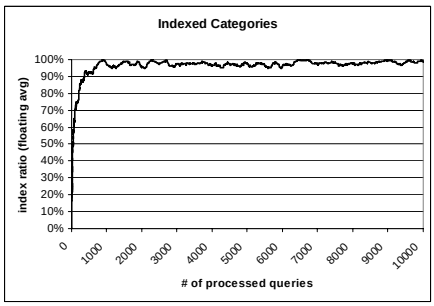


Fig. 3. Coverage of category index

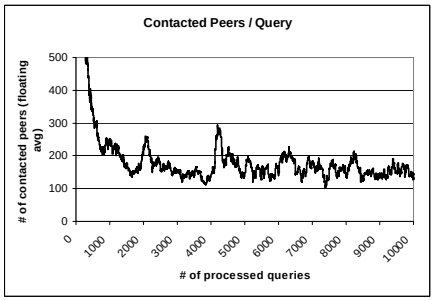


Fig. 4. Contacted peers per query

popularity shifts on the whole outcome can also be seen clearly. The category index takes care that the peaks caused by popularity shifts don't become too high. Summarized, the combination of both indexes yields a high decrease of contacted peers compared to broadcasting.

In the experiments described here we didn't introduce dynamics regarding the peers contents. Therefore, our algorithm yields exactly the same results as a complete index. In [12] (where only keyword preferences are taken into account), we show that if 20% of the peers contents change during a simulation run, the error ratio is about 3.5%.

Chapter 7

Summary and Future Work

7.1 Summary

With the success of the Web, integration of distributed information has become one of the most important research areas in computer science. It is clear that there is no 'one-size-fits-all' solution to this field. Therefore, we have focused our work to the context of digital library networks, and have identified requirements for search in such networks. As result of an extensive review of related work, design dimensions of schema-based peer-to-peer networks have been distilled, and the existing approaches have been classified according to these dimensions[1].

The main contribution of this thesis is a comprehensive proposal for a peer-to-peer infrastructure which allows to efficiently search for documents in such a network, based on their metadata and content. Specifically, the following achievements can be identified:

SPQR Algorithm As foundation, we devised the SPQR algorithm for efficient query distribution in a digital library network. SPQR builds on ideas from distributed databases, where central mediators collect catalog data from network nodes. Based on this information, they create distributed plans for incoming queries and coordinate their execution. In SPQR, this idea is developed further from a central mediator to a super-peer mediator network, where no central coordination and information is required anymore. We have shown the correctness of SPQR; for any query, it retrieves the same result set as if the query would have been executed on a central database containing the content of all nodes in the network.

OAI-P2P Infrastructure A complete infrastructure has been designed and implemented which realizes the SPQR algorithm. This infrastructure is based on the Semantic Web data

[1] The proposed design dimensions already have been adopted in a survey of search in P2P networks [146]

representation format RDF and a query exchange language derived from datalog. Wrappers to different backends (such as RDBMSs and Semantic Web data stores) are available to connect a wide variety of information providers to the network. To cater specifically for digital library systems, an OAI wrapper has been developed which allows to add any digital library with an OAI-PMH conformant interface to the network, without imposing any additional requirements on the digital library system.

Preference Query Support Experience with the implemented infrastructure showed that support for hard-constraints only (as in traditional database queries) is not sufficient, and the notion of soft-constraints to express best matches needs to be added. Therefore we have extended the basic SPQR algorithm to support the expression of soft-constraints. We chose preference queries as underlying formalism, because they are currently the most versatile means to express best match notions, covering other approaches such as scoring and top-k queries. We have shown that correctness of distributed query evaluation also holds for this extended case. Also we have proposed several strategies to optimize query distribution further, by introducing the concept of query-driven index creation and maintenance. This concept avoids any index maintenance messages, and is suitable for contexts where query frequencies follow a skewed, Zipfian distribution.

Simulation-based Evaluation To evaluate our algorithms, we have implemented a flexible and scalable Simulation framework for schema-bases P2P networks. We have conducted extensive experiments with respect to the influence of peer clustering on query distribution and load-balancing within the super-peer backbone. The preference query support has been evaluated based on a large TREC document collection, and the proposed algorithm has turned out to be efficient as well as effective under network churn.

7.2 Future Work

Extended Query Planning Currently, SPQR is only able to handle horizontally fragmented data. While this is the predominant kind of fragmentation in the digital library context, it is still worthwhile to consider vertical fragmentation as well. First steps in this direction have already been made [27]. The main challenge here is to identify for a given complex query the super-peer nodes where splitting off sub-queries makes most sense, without provision of global knowledge.

Self-Organized Index/Cache Optimization SPQR as well as its extension to preference queries offer several degrees of freedom with respect to index creation maintenance. Currently, the actual choices for index creation and maintenance are configured at start-up time of super-peers. This is an area where self-organization would improve index efficiency:

- *Index Granularity* A better approach than hard-wired index granularity for each attribute would be to decide based on usage in queries and actual size of the value domain in the network. For example, the each super-peer could autonomously find out that the *dc:language* attribute has a domain of very limited size, and that queries often contain constraints to exactly one value of this attribute. That would lead to the creation of an index for *dc:language* with value granularity without having to encode this behavior in advance.

- *Indexing vs. Caching* Instead of forwarding all queries to its peers, super-peers may also start to cache result sets for most frequent (sub-)queries. Work on this optimization has already begun [26], and it has turned out that caching can be worthwhile for Zipfian query frequency distributions.

- *Index Maintenance Strategy* Currently we use different index maintenance strategies for basic SQPR and SPQR with preference query support. For the former, index content is collected and maintained in advance, whenever the network content changes. For the latter, we use query-driven maintenance, maintaining index entries only on demand. It is already known that maintaining a complete inverse document frequency index in advance is too costly. But it probably is advantageous to introduce query-driven updates also for the basic SPQR indexes, especially for finer granularities where up-front maintenance of a full index would be too expensive.

- *Index Update Rate* The index update rate is determined by two opposing factors: On the one hand we want to update indices as soon as possible, to increase correctness of query results. On the other hand we want to update them as late as possible, to decrease index maintenance effort. The optimal update rate depends on content change rate in the network, and is varying for different attributes. For example, new documents may be added frequently to peers, therefore the term index might need a short index entry expiration period. But peers providing content in just one language typically continue to do so over larger time spans, therefore, the language attribute index can be updated very lazily without loss of query result precision.

All these decisions would be based on statistics gathered with respect to query frequency, content distribution, and network churn. Instead of hard-coding index maintenance decisions, super-peers would be equipped with analytical error and effort models, which would allow them to estimate cost and benefit of a specific index maintenance decision, and choose the

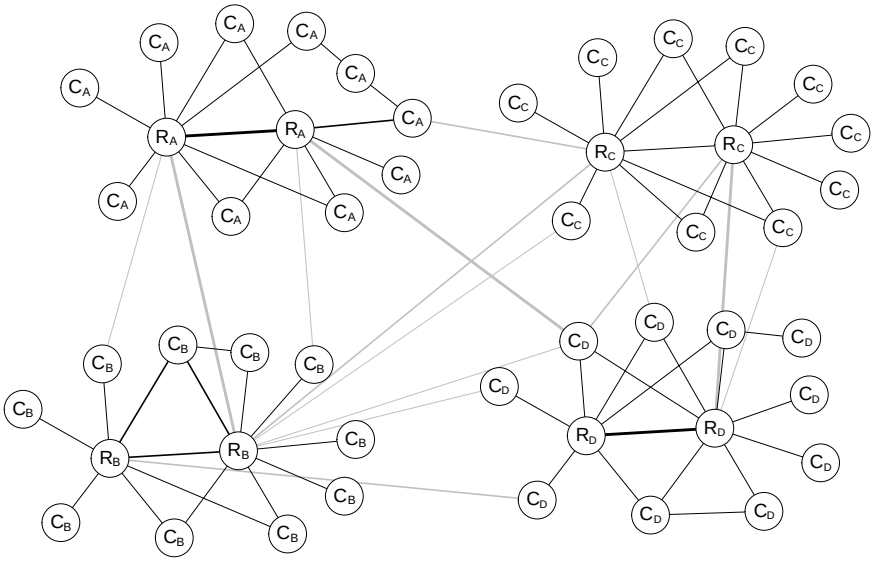


Figure 7.1: Recommender and content peers

optimal trade-off. First results of such a model for error estimation can be found in [165].

Other topologies The introduction of super-peers which take over responsibility for query routing has been shown to be advantageous. However, the current choice of a strict two-level classification of nodes in peers and super-peers may be a bit too rigid. With respect to peer capabilities, we do not really have a two-class-society, but rather a continuous spectrum of more or less powerful peers. Therefore, for an optimal utilization of all peers abilities, we should strive for a model where peers grow gradually into the super-peer role. Such a model is only possible when giving up a structured, highly symmetric super-peer topology. We are considering instead a short-cut topology (such as [105]), but with a distribution of outgoing links according to each peer's characteristics (bandwidth, storage, processing power). Figure 7.1 shows a possible topology. Here, more powerful peers gradually take over the recommender role (denoted by 'R'). Queries are routed first to the most suitable recommender, then along recommender connections until they reach the appropriate content providers.

This approach requires a clustering strategy even more than the current SPQR topology, because otherwise it would degenerate into a Gnutella-like network. On the other hand, good optimization policies may make it possible to let the network topology approximate a small-world graph over time. We are currently investigating the use of Bloom filters as peer self-descriptions to facilitate such a self-organized clustering [137].

Appendices

Appendix A

Publications

Journal articles

1. Combining ontologies and peer-to-peer technologies for inter-organizational knowledge management.
   The Learning Organization 12(5), 2005, pp. 480-491.
   Co-authors: Heiner Stuckenschmidt, Wolfgang Nejdl.
2. Super-peer-based routing strategies for RDF-based peer-to-peer networks.
   Web Semantics 1 (2), Feb. 2004, pp. 137-240.
   Co-authors: Wolfgang Nejdl, Martin Wolpers, Christoph Schmitz, Mario Schlosser, Ingo Brunkhorst, Alexander Löser.
3. Design issues and challenges for RDF- and schema-based peer-to-peer systems.
   SIGMOD Record, September 2003, pp. 41-46.
   Co-authors: Wolfgang Nejdl, Michael Sintek

Conference publications

4. Querying the Semantic Web with preferences.
   In *Proceedings of the 5th International Semantic Web Conference (ISWC)*,
   Athens, GA, USA, 2006
   Co-authors: Jeff Z. Pan, Uwe Thaden.
5. DL meets P2P – Distributed document retrieval based on classification and content.
   In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, Vienna, Austria, 2005. Best paper award.
   Co-authors: Wolf-Tilo Balke, Wolfgang Nejdl, Uwe Thaden.
6. Progressive distributed top-k retrieval in peer-to-peer networks.
   In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*,
   Tokyo, Japan, 2005.
   Co-authors: Wolf-Tilo Balke, Wolfgang Nejdl, Uwe Thaden.

7. Top-k query evaluation for schema-based peer-to-peer networks.
   In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*,
   Hiroshima, Japan, 2004.
   Co-authors: Wolfgang Nejdl, Uwe Thaden, Wolf-Tilo Balke.

8. Information integration in schema-based peer-to-peer networks.
   In *Proceedings of the 15th Conference on Advanced Information Systems Engineering (CAiSE)*, Klagenfurt/Velden, Austria, 2003
   Co-authors: Alexander Löser, Martin Wolpers, Wolfgang Nejdl.

9. Towards the systematic use of interfaces in Java programming.
   In *Proceedings of the 2nd International Conference on the Principles and Practice of Programming in Java (PPPJ)*, Kilkenny City, Ireland, 2003.
   Co-authors: Friedric Steimann, Thomas Kühne

10. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks.
    In *Proceedings of the 12th International World Wide Web Conference (WWW)*,
    Budapest, Hungary, 2003.
    Co-authors: Wolfgang Nejdl, Martin Wolpers, Christoph Schmitz, Mario Schlosser, Ingo Brunkhorst, Alexander Löser.

11. Towards a modification exchange language for distributed RDF repositories.
    In *Proceedings of the International Semantic Web Conference (ISWC)*,
    Sardinia, Italy, 2002.
    Co-authors: Wolfgang Nejdl, Bernd Simon, Julien Tane.

12. Animiertes UML als Medium für die Didaktik der objektorientierten Programmierung.
    In *Proceedings of Modellierung 2002*, Tutzing, Germany, 2002.
    Co-authors: Friedrich Steimann, Uwe Thaden, Wolfgang Nejdl.

Workshop publications

13. Search strategies for scientific collaboration networks.
    In *Proceedings of the Workshop on Information Retrieval in Peer-to-Peer-Networks, ACM Fourteenth Conference on Information and Knowledge Management (CIKM)*,
    Bremen, Germany, 2005.
    Co-authors: Paul - Alexandru Chirita, Andrei Damian, Wolfgang Nejdl.

14. Caching for improved retrieval in peer-to-peer networks.
    In: *Proceedings of the GI/ITG-Workshop Peer-to-Peer-Systeme und -Anwendungen*,
    Kaiserslautern, Germany, 2005.
    Co-authors: Wolf-Tilo Balke, Wolfgang Nejdl, Uwe Thaden

15. A simulation framework for schema-based query routing in P2P networks.
    In *Proceedings of the Workshop on Peer-to-Peer Computing and Databases in conjunction with EDBT*, Heraklion, Greece, 2004.
    Co-author: Uwe Thaden.

16. Efficient data store discovery in a scientific P2P network.
    In *Proceedings of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, International Semantic Web Conference (ISWC)*,
    Sunibel Island, Florida, USA, 2003.
    Co-authors: Alexander Löser, Martin Wolpers, Wolfgang Nejdl.

17. Semantic overlay clusters within super-peer networks.
    In *Proceedings of the International Workshop on Databases, Information Systems, and P2P Computing, colocated with 29th International Conference on Very Large Databases (VLDB)*, Berlin, Germany, 2003.
    Co-authors: Alexander Löser, Felix Naumann, Wolf Siberski, Wolfgang Nejdl, Uwe Thaden.

18. OAI-P2P: A peer-to-peer network for Open Archives.
    In *Proceedings of the Workshop on Distributed Computing Architectures for Digital Libraries, 31st Intl. Conference on Parallel Processing (ICPP)*,
    Vancouver, Canada, 2002.
    Co-authors: Benjamin Ahlborn, Wolfgang Nejdl.

Book contributions

19. Peer-to-peer and Semantic Web.
    In *Steffen Staab and Heiner Stuckenschmidt (Editors).Semantic Web and Peer-to-Peer*,
    Springer, 2005.
    Co-authors: Heiner Stuckenschmidt, Frank van Harmelen, Steffen Staab.

20. Schema-based peer-to-peer systems.
    In *Ralf Steinmetz and Klaus Wehrle (Editors). Peer-to-Peer Systems and Applications.*
    Springer, 2005.
    Co-author: Wolfgang Nejdl.

21. Object-oriented construction handbook.
    Morgan Kaufman Publishers, 2005.
    Main author: Heinz Züllighoven; other lead authors: Robert F. Beeger, Wolf-Gideon Bleek, Guido Gryczan, Carola Lilienthal, Martin Lippert, Stefan Roock, Thomas Slotos, Dirk Weske, Ingrid Wetzel.

22. Role Object.
    In *Neil Harrison, Brian Foot, Hans Rohnert (Editors). Pattern Languages of Program Design 4*, Addison-Wesley, 2000.
    Co-authors: Dirk Bäumer, Dirk Riehle, Martina Wulf.

23. Serializer.
    In *Robert Martin, Dirk Riehle, Frank Buschmann (Editors). Pattern Languages of Program Design 3*, Addison-Wesley, 1998.
    Co-authors: Dirk Riehle, Dirk Bäumer, Daniel Megert, Heinz Züllighoven.

Technical Reports

24. On Bloom Filters.
    Technical Report, 2006. http://www.l3s.de/ papapetrou/publications/onbloomfilters.pdf.
    Co-author: Odysseas Papapetrou and Wolfgang Nejdl.

25. Estimating Index Staleness under Network Churn.
    Technical Report, 2006. http://www.l3s.de/ siberski/reports/index-staleness.pdf.
    Co-author: Uwe Thaden and Wolfgang Nejdl.

26. Edutella Query Exchange Language specification.
    Technical Report, 2004. http://edutella.jxta.org/spec/qel.html.
    Co-author: Mikael Nilsson.

27. Edutella Retrieval Service specification.
    Technical Report, 2004. http://edutella.jxta.org/spec/qel.html.
    Co-authors: Mikael Nilsson, Julien Tane.

28. A Semantic-Web based peer-to-peer service registry network.
    Technical Report, 2003. Co-authors: Uwe Thaden, Wolfgang Nejdl.

List of Figures

List of Tables

Bibliography

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: a self-organizing structured P2P system. *SIGMOD Record*, 32(3):29–33, 2003.

[2] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. Start making sense: The Chatty Web approach for global semantic agreements. *Web Semantics*, 1(1), 2003.

[3] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In *Proceedings of the Second International Semantic Web Conference (ISWC)*, Sanibel Island, FL, USA, 2003.

[4] K. Aberer, A. Datta, M. Hauswirth, and R. Schmidt. Indexing data-oriented overlay networks. In *Proceedings of 31st International Conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005.

[5] L. A. Adamic and B. A. Huberman. Zipf's law and the internet. *Glottometrics*, 3, 2002. http://ginger.hpl.hp.com/shl/papers/ranking/adamicglottometrics.pdf.

[6] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Phy. Rev. E*, 64:46135–46143, Oct. 2001.

[7] B. Ahlborn, W. Nejdl, and W. Siberski. OAI-P2P: A peer-to-peer network for open archives. In *Proceedings of the Workshop on Distributed Computing Architectures for Digital Libraries*, Vancouver, Canada, August 2002.

[8] Anglo-american cataloguing rules, 2002. http://www.aacr2.org.

[9] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System R: Relational approach to database management. *ACM Trans. Database Syst.*, 1(2):97–137, 1976.

[10] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press, Cambridge, 2003.

[11] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. DL meets P2P – distributed document retrieval based on classification and content. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, Vienna, Austria, 2005.

[12] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceeding of Proceedings of the 21ar International Conference on Data Engineering (ICDE)*, Tokyo, Japan, 2005.

[13] D. Beckett. RDF/XML syntax specification, 2004. W3C Recommendation, http://www.w3.org/TR/rdf-syntax-grammar/.

[14] S. Bergamaschi, F. Guerra, and M. Vincini. A peer-to-peer information system for the semantic web. In *Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing*, Melbourne, Australia, 2003.

[15] T. Berners-Lee. Notation 3. Technical report, World Wide Web Consortium, 1998. Design Note, http://www.w3.org/DesignIssues/Notation3.

[16] T. Berners-Lee. RDF and the semantic web, 2000. Slides of keynote speech at XML2000, available at http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html.

[17] T. Berners-Lee. N3QL – RDF data query language, 2004. http://www.w3.org/DesignIssues/N3QL.html.

[18] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[19] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 353–366, Portland, Oregon, USA, 2004.

[20] H. Boley, S. Tabet, and G. Wagner. Design rationale for RuleML: A markup language for semantic web rules. In *Proceedings of the first Semantic Web Working Symposium*, pages 381–401, Stanford University, California, USA, 2001.

[21] M. Borghuis, H. Brinckman, A. Fischer, K. Hunter, E. van der Loo, R. ter Mors, P. Mostert, and J. Zijlstra. TULIP final report: The university licensing program, 1996. Available at http://web.archive.org/web/20000208135700/www1.elsevier.com/homepage/about/resproj/trmenu.htm.

[22] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.

[23] C. L. Borgmann. *From Gutenberg to the Global Information Infrastructure: Access to Information in the Networked World*. MIT Press, Cambridge, MA, USA, 2000.

[24] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Proceedings of the Second International Semantic Web Conference (ISWC)*, pages 164–179, Sanibel Island, FL, USA, 2003.

[25] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, S. Seltzsam, and K. Stocker. Objectglobe: Open distributed query processing services on the internet. *IEEE Data Eng. Bull.*, 24(1):64–70, 2001.

[26] I. Brunkhorst and H. Dhraief. Semantic caching in schema-based peer-to-peer networks. In *Proceedings of the Workshop on Databases, Information Systems and P2P (DBISP2P)*, Trondheim, Norway, 2005.

[27] I. Brunkhorst, H. Dhraief, A. Kemper, W. Nejdl, and C. Wiesner. Distributed queries and query optimization in schema-based peer-to-peer systems. In *International Workshop on Databases, Information Systems, and P2P Computing, colocated with 29th International Conference on Very Large Databases*, Berlin, Germany, September 2003.

[28] F. Bry and S. Schaffert. The XML query language Xcerpt: Design principles, examples, and semantics. In *Proceedings of the International Workshop on Web and Databases*, 2002.

[29] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

[30] M. Cai and M. R. Frank. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In *Proceedings of the 13th international conference on World Wide Web (WWW 2004)*, pages 650–657, New York, NY, USA, 2004.

[31] M. Cai, M. R. Frank, J. Chen, and P. A. Szekely. MAAN: A multi-attribute addressable network for grid information services. *J. Grid Comput.*, 2(1):3–14, 2004.

[32] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05)*, Boston, MA, USA, 2005.

[33] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrstructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, October 2002.

[34] D. D. Chamberlin and R. F. Boyce. SEQUEL: A structured english query language.

In *FIDET '74: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264, New York, NY, USA, 1974. ACM Press.

[35] Q. Chen. The origin of power laws in internet topologies revisited. In *21st Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[36] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[37] J. Chomicki. Datalog, 2004. http://www.cse.buffalo.edu/ chomicki/635/a1.pdf.

[38] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An efficient SQL-based RDF querying scheme. In *Proceedings of 31st International Conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, 2005.

[39] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[40] L. S. Connaway, B. F. Lavoie, and E. T. O'Neill. Mining for digital resources: Identifying and characterizing digital materials in WorldCat. In *Proceedings of the ACRL 12th National Conference Wide Web*, 2005.

[41] Copac: Academic & national library catalogue. http://copac.ac.uk/.

[42] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulations. In *International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, USA, Jun 1999.

[43] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings International Conference on Distributed Computing Systems*, July 2002.

[44] A. Crespo and H. G. Molina. Semantic overlay networks for P2P systems. Technical report, Stanford University, 2003.

[45] R. Cyganiak. A relational algebra for SPARQL. Technical Report HPL-2005-170, Hewlett-Packard Labs, 2005.

[46] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing peer-to-peer systems. In *Proceedings of the 9th International Conference on Database Theory (ICDT2003)*, pages 1–15, Siena, Italy, 2003.

[47] DCMI Usage Board. DCMI metadata terms. Technical report, Dublin Core Metadata Initiative, 2005. http://dublincore.org/documents/2005/06/13/dcmi-terms/.

[48] M. Dean and G. Schreiber. OWL Web Ontology Language reference, 2004. http://www.w3.org/TR/owl-ref/.

[49] M. Deegan and S. Tanner. *Digital futures: Strategies for the information age*. Facet Publishing, London, UK, 2002.

[50] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in distributed e-learning environments. In *Proceedings of the 13th international conference on World Wide Web*, pages 170–179, New York, NY, USA, 2004.

[51] Edutella homepage and source code, 2005. http://edutella.jxta.org.

[52] M. Ehrig, C. Schmitz, S. Staab, J. Tane, and C. Tempich. Towards evaluation of peer-to-peer-based distributed information management systems. In L. van Elst et al., editors, *Agent-mediated Knowledge Management - AMKM-2003, AAAI Spring Symposium 2003, Stanford University, March 24-26*, 2003.

[53] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Santa Barbara, California, USA, 2001.

[54] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review, Vol 29(4)*, 1999.

[55] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. Strudel: A web-site management system. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 549–552, Tucson, Arizona, USA, 1997.

[56] E. Franconi, G. M. Kuper, A. Lopatenko, and I. Zaihrayeu. The coDB robust peer-to-peer database system. In *Proceedings of the Twelfth Italian Symposium on Advanced Database Systems (SEBD 2004)*, pages 382–393, S. Margherita di Pula, Cagliari, Italy, 2004.

[57] M. J. Franklin, B. T. Jónsson, and D. Kossmann. Performance tradeoffs for client-server query processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 149–160, Montreal, Canada, 1996.

[58] H. Garcia-Molina, J. D. Ullmann, and J. Widom. *Database Systems*. Prentice Hall, Upper Saddle River, New Jersey, 2002.

[59] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, 2002.

[60] C. L. Giles, K. D. Bollacker, and S. Lawrence. CiteSeer: an automatic citation indexing

system. In *Proceedings of the third ACM conference on Digital libraries (DL'98)*, pages 89–98, Pittsburgh, PA, USA, 1998. ACM Press.

[61] Gnutella protocol specification v0.4. www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[62] Gnutella 0.6 protocol draft. http://rfc-gnutella.sourceforge.net/developer/testing/index.html.

[63] Google Scholar homepage. http://scholar.google.com/.

[64] G. Graefe. Volcano – an extensible and parallel query evaluation system. *IEEE Trans. Knowl. Data Eng.*, 6(1):120–135, 1994.

[65] G. Graefe. The Cascades framework for query optimization. *IEEE Data Eng. Bull.*, 18(3):19–29, 1995.

[66] S. Gribble, A. Y. Halevy, Z. G. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proceedings of the Fourth International Workshop on the Web and Databases (WebDB '2001).*, Santa Barbara, CA, USA, May 2001.

[67] K. Hagedorn. OAIster: a "no dead ends" OAI service provider. *Library Hi Tech*, 21(2):170–181, 2003.

[68] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza peer data management system. *IEEE Trans. Knowl. Data Eng.*, 16(7):787–798, 2004.

[69] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, Budapest, Hungary, May 2003.

[70] T. Hargreaves. The FastTrack protocol, 2004. http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?rev=HEAD&content-type=text/vnd.viewcvs-markup.

[71] R. Heese, S. Herschel, F. Naumann, and A. Roth. Self-extending peer data management. In *Proceedings of Datenbanksysteme in Business, Technologie und Web, 11. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (BTW2005)*, pages 165–174, Karlsruhe, Germany, 2005.

[72] Hewlett Packard Research Labs. RDQL - RDF data query language, 2004. http://www.hpl.hp.com/semweb/rdql.html.

[73] P. Hitzler, J. Angele, B. Motik, and R. Studer. Bridging the paradigm gap with rules for

OWL. In *Proceedings of the W3C Workshop on Rule Languages for Interoperability*, Washington, DC, USA, 2005.

[74] R. Huebsch, B. N. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The architecture of PIER: an internet-scale query processor. In *Proceedings of the Second Biennial Conference on Innovative Data Systems Research*, pages 28–43, Asilomar, CA, USA, 2005.

[75] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the internet with PIER. In *Proceedings of 29th International Conference on Very Large Data Bases*, pages 321–332, Berlin, Germany, 2003.

[76] Y. Ioannidis, D. Maier, S. Abiteboul, P. Buneman, S. Davidson, E. Fox, A. Halevy, C. Knoblock, F. Rabitti, H. Schek, and G. Weikum. Digital library information-technology infrastructures. *Int J Digit Libr*, 5(4):266 – 274, 2005.

[77] Y. E. Ioannidis and E. Wong. Query optimization by simulated annealing. In *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data Annual Conference (SIGMOD*, pages 9–22, San Francisco, CA, USA, 1987.

[78] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonom. In *Proceedings of the 10th International Conference on Research in Computational Linguistics (ROCLING X)*, pages 19–33, Taipei, Taiwan, 1997.

[79] S. Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Proceedings of the NETWORKING 2002 Workshop on Web Engineering and Peer-to-Peer Computing*, pages 202–214, Pisa, Italy, 2002.

[80] V. Josifovski and T. Risch. Integrating heterogenous overlapping databases through object-oriented transformations. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 435–446, Edinburgh, Scotland, UK, 1999. Morgan Kaufmann.

[81] V. Josifovski, P. Schwarz, L. Haas, and E. Lin. Garlic: A new flavor of federated query processing for DB2. In *In Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD2002)*, pages 524–532, Madison, WI, USA, June 2002.

[82] M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Second International Workshop on Peer-to-Peer Systems*, pages 98–107, Berkeley, CA, USA, 2003.

[83] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL:

A declarative query language for RDF. In *The 11th. Intnational World Wide Web Conference*, 2002.

[84] W. Kießling. Foundations of preferences in database systems. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 311–322, Hong Kong, China, 2002.

[85] G. Kokkinidis and V. Christophides. Semantic query routing and processing in P2P database systems: The ICS-FORTH SQPeer middleware. In *Proceedings of the EDBT 2004 Workshop on Peer-to-Peer Computing and Databases*, pages 486–495, Heraklion, Crete, Greece, 2004.

[86] D. Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[87] T. Krichel and S. M. Warner. Academic self-documentation: which way forward for computing, library and information science? In *Proceedings of the 4th International Conference of Asian Digital Libraries*, Bangalore, India, 2001.

[88] J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton, D. Geels, R. Gummadi, S. C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Y. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 190–201, Cambridge, MA, USA, 2000.

[89] Karlsruher virtueller katalog. http://www.ubka.uni-karlsruhe.de/hylib/virtueller_katalog.html.

[90] C. Lagoze and J. R. Davis. Dienst: an architecture for distributed document libraries. *Commun. ACM*, 38(4):47, 1995.

[91] C. Lagoze and H. Van de Sompel. The Open Archives Initiative: building a low-barrier interoperability framework. In *Proceedings of ACM/IEEE Joint Conference on Digital Libraries (JCDL2001)*, Roanoke, VA, USA, 2001.

[92] C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. The Open Archives Initiative protocol for metadata harvesting, 2002. http://www.openarchives.org/OAI/openarchivesprotocol.html.

[93] R. S. G. Lanzelotte, P. Valduriez, and M. Zaït. On the effectiveness of optimization search strategies for parallel execution spaces. In *Proceedings of the 19th International Conference on Very Large Data Bases (VLDB)*, pages 493–504, Dublin, Ireland, 1993.

[94] M. Lesk. The digital library: What is it? Why should it be here?, De-

cember 1992. Summary of the Workshop on Digital Libraries, available at http://fox.cs.vt.edu/DigitalLibrary/DLSBch3.ps.

[95] M. Lesk. Workshop on electronic libraries, July 1992. Summary of the Workshop on Electronic Libraries, available at http://fox.cs.vt.edu/DigitalLibrary/DLSBch2.ps.

[96] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of 22th International Conference on Very Large Data Bases*, pages 251–262, Mumbai (Bombay), India, 1996. Morgan Kaufmann.

[97] D. M. Levy and C. C. Marshal. Going digital: a look at assumptions underlying digital libraries. *CACM*, 38(4):77–84, 1995.

[98] C. Li, M. A. Soliman, K. C.-C. Chang, and I. F. Ilyas. RankSQL: Supporting ranking queries in relational database management systems. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 1342–1345, Trondheim, Norway, 2005.

[99] Y. Li, Z. A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knwoledge and Data Engineering*, 15(4), 2003.

[100] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 296–304, San Francisco, CA, USA, 1998.

[101] X. Liu, K. Maly, M. Zubair, and M. L. Nelson. Arc: an OAI service provider for cross-archive searching. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 2001.

[102] P. Lockemann, U. Kölsch, A. Koschel, R. Kramer, R. Nikolai, M. Wallrath, and H.-D. Walter. The network as a global database: Challenges of interoperability, proactivity, interactiveness, legacy. In *Proceedings of the 23rd Conference on Very Large Data Bases (VLDB)*, pages 567–574, Athens, Greece, Aug 1997.

[103] G. M. Lohman, C. Mohan, L. M. Haas, D. Daniels, B. G. Lindsay, P. G. Selinger, and P. F. Wilms. Query processing in R*. In *Query Processing in Database Systems*, pages 31–47. Springer, 1985.

[104] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The case for a hybrid P2P search infrastructure. In *Proceedings of the Third International Workshop on Peer-to-Peer Systems (IPTPS)*, La Jolla, CA, USA, 2004.

[105] A. Löser, C. Tempich, B. Quilitz, W.-T. Balke, S. Staab, and W. Nejdl. Searching dynamic communities with personal indexes. In *Proceedings of the 4th International SemanticWeb Conference (ISWC)*, pages 491–505, Galway, Ireland, 2005.

[106] A. Löser, M. Wolpers, W. Siberski, and W. Nejdl. Efficient data store discovery in a scientific P2P network. In *Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, International Semantic Web Conference (ISWC 2003)*, Sunibel Island, FL, USA, 2003.

[107] Lucene homepage. http://lucene.apache.org/java.

[108] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing (ICS)*, pages 84–95, 2002.

[109] A. Magkanaraki, V. Tannen, V. Chistophides, and D. Plexousakis. Viewing the semantic web through RVL lenses. In *Proceedings of the Second International Semantic Web Conference*, Sanibel Island, FL, USA, 2003.

[110] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*, pages 183–192, 2002.

[111] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Seattle, Washington, USA, 2003.

[112] MARC21 format for bibliographic data, 1999. http://www.loc.gov/marc.

[113] J. M. Martínez. MPEG-7 overview (version 10), 2004. ISO/IEC JTC1/SC29/WG11 (Coding of Moving Pictures and Audio) Document N6828, available at http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

[114] B. McBride. Jena: A semantic web toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.

[115] S. H. McCallum. MARC: keystone for library automation. *IEEE Annals of the History of Computing*, 24(2):34–49, 2002.

[116] A. Medina, I. Matta, and J. Byers. On the origin of power laws in internet topologies. *ACM SIGCOMM Computer Communication Review*, 30(2), 2000.

[117] The Mercury project and library information system II – the first three years, 1992. Available at http://www.cs.cornell.edu/wya/papers/Mercury6.doc.

[118] S. Meregu, S. Srinivasan, and E. Zegura. Adding structure to unstructured peer-to-peer networks: the use of small-world graphs. *J Parallel Distrib Comput.*, 65:142–153, 2005.

[119] Z. Miklós, G. Neumann, U. Zdun, and M. Sintek. Querying semantic web resources using TRIPLE views. In *Proceedings of the Dagstuhl Seminar on Semantic Interoperability and Integration*, 2005.

[120] A. Mondal, K. Goda, and M. Kitsuregawa. Effective load-balancing via migration and replication in spatial grids. In *Proceedings of the 14th International Conference on Database and Expert Systems Applications (DEXA 2003)*, Prague, Czech Republic, Sep 2003.

[121] J. Mullin. Optimal semijoins for distributed database systems. *IEEE Transactions on Software Engineering*, 16(5):558–560, 1990.

[122] M. Naor and U. Wieder. A simple fault tolerant distributed hash table. In *Second International Workshop on Peer-to-Peer Systems*, pages 88–97, Berkeley, CA, USA, 2003.

[123] Napster. http://www.napster.com/.

[124] W. Nejdl, W. Siberski, and M. Sintek. Design issues and challenges for RDF- and schema-based peer-to-peer systems. *SIGMOD Record*, May 2003.

[125] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: a P2P Networking Infrastructure based on RDF. In *Proceedings of the Eleventh International World Wide Web Conference (WWW2002)*, Hawaii, USA, May 2002.

[126] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser. Super-peer-based routing strategies for RDF-based peer-to-peer networks. *Web Semantics*, 1(7):137–240, 2005.

[127] W. Nejld, W. Siberski, U. Thaden, and W.-T. Balke. Top-k query evaluation for schema-based peer-to-peer networks. In *Proceedings of 3rd International Semantic Web Conference (ISWC)*, 2004.

[128] H. Neuroth and T. Pianos. Vascoda: A german scientific portal for cross-searching distributed digital resource collections. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 257–262, Trondheim, Norway, 2003.

[129] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. PeerDB: A P2P-based system for dis-

tributed data sharing. In *Proceedings of the 19th International Conference on Data Engineering*, pages 633–644, Bangalore, India, 2003.

[130] M. Nilsson and W. Siberski. RDF query exchange language (QEL) – concepts, semantics and RDF syntax. Technical report, Edutella Project, 2004. http://edutella.jxta.org/spec/qel.html.

[131] Personalized access to distributed learning repositories – 2004 year report, 2004. http://www.swedishlearninglab.org/documents/PADLR-final-report.pdf.

[132] M. Palmér and A. Naeve. Conzilla – a conceptual interface to the semantic web. In *Proceedings of the 13th International Conference on Conceptual Structures (ICCS)*, pages 136–151, Kassel, Germany, 2005.

[133] M. Palmér, A. Naeve, and F. Paulsson. The SCAM framework: Helping semantic web applications to store and access metadata. In *Proceedings of the 1st European Semantic Web Symposium (ESWS)*, Heraklion, Greece, 2004.

[134] J. Z. Pan and I. Horrocks. RDFS(FA) and RDF MT: Two semantics for RDFS. In *Proceedings of the Second International Semantic Web Conference*, pages 30–46, Sanibel Island, FL, USA, 2003.

[135] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 467–478, San Diego, CA, USA, 2003.

[136] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE)*, pages 251–260, Taipei, Taiwan, 1995.

[137] O. Papapetrou, W. Siberski, and W. Nejdl. On bloom filters, 2006. Available at http://www.l3s.de/ papapetrou/publications/onbloomfilters.pdf.

[138] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, Newport, RI, USA, 1997.

[139] E. Prud'hommeaux and B. Grosof. RDF query survey, 2001. http://www.w3.org/2001/11/13-RDF-Query-Rules/.

[140] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF, 2006. W3C Candidate Recommendation, http://www.w3.org/TR/rdf-sparql-query/.

[141] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content

addressable network. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press New York, NY, USA, 2001.

[142] W. B. Rayward. A history of computer applications in libraries: prolegomena. *IEEE Annals of the History of Computing*, 24(2):4–15, 2002.

[143] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res. (JAIR)*, 11:95–130, 1999.

[144] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, Boston, MA, USA, 2004.

[145] D. Riecken. Introduction: personalized views of personalization. *Commun. ACM*, 43(8):26–28, 2000. (Introduction to Special Issue on Personalization).

[146] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. Technical report, University of New South Wales, Sydney, Australia, 2004. Technical Report UNSW-EE-P2P-1-1, accepted to appear in *Computer Networks*.

[147] J. Ritter. Why Gnutella can't scale. Technical report, Darkridge, Inc., 2001. http://www.darkridge.com/~jpr5/doc/gnutella.html.

[148] J. Robie, L. M. Garshol, S. Newcomb, M. Biezunski, M. Fuchs, L. Miller, D. Brickley, V. Christophides, and G. Karvounarakis. The syntactic web. *Markup Lang.*, 3(4):411–440, 2001.

[149] M. Röscheisen, M. Q. W. Baldonado, K. C.-C. Chang, L. Gravano, S. P. Ketchpel, and A. Paepcke. The Stanford InfoBus and its service layers: Augmenting the internet with high-level information management protocols. In *Digital Libraries in Computer Science: The MeDoc Approach*, pages 213–230, 1998.

[150] M. Roussopoulos, M. Baker, D. S. H. Rosenthal, T. J. Giuli, P. Maniatis, and J. C. Mogul. 2 P2P or not 2 P2P? In *Proceedings of the Third International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 33–43, La Jolla, CA, USA, 2004.

[151] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, Heidelberg, Germany, 2001.

[152] Regeln für den Schlagwortkatalog (RSWK), 1998. 3rd edition.

[153] Rule interchange format working group charter, 2005. http://www.w3.org/2005/rules/wg/charter.

[154] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN)*, January 2002.

[155] N. Sarshar and V. P. Roychowdhury. Scale-free and stable structures in complex ad hoc networks. *Phys. Rev. E*, 69, 2004. no. 026101.

[156] C. Sartiani, P. Manghi, G. Ghelli, and G. Conforti. XPeer: A self-organizing XML P2P database system. In *Proceedings of the EDBT 2004 Workshop on Peer-to-Peer Computing and Databases*, pages 456–465, Heraklion, Crete, Greece, 2004.

[157] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks. In *International Workshop on Agents and Peer-to-Peer Computing*, Bologna, Italy, July 2002.

[158] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A scalable and ontology-based P2P infrastructure for semantic web services. In *Proceedings of the Second International Conference on Peer-to-Peer Computing*, Linköping, Sweden, September 2002.

[159] C. Schmitz. Self-organizing a small world by topic. In *Proceedings of the MobiQuitous'04 Workshop on Peer-to-Peer Knowledge Management (P2PKM 2004)*, Boston, MA, USA, 2004.

[160] Scirus service provider, 2001. http://www.scirus.com/html/scirus_service_provider.htm.

[161] L. Serafini, A. Borgida, and A. Tamilin. Aspects of distributed and modular ontology reasoning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005.

[162] L. Serafini, F. Giunchiglia, J. Mylopoulos, and P. A. Bernstein. Local relational model: A logical formalization of database coordination. In *Proceedings of the 4th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT)*, pages 286–299, Stanford, CA, USA, 2003.

[163] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, 1990.

[164] W. Siberski and U. Thaden. A simulation framework for schema-based query routing in P2P networks. In *1st International Workshop on Peer-to-Peer Computing & DataBases(P2P& DB 2004*, 2004.

[165] W. Siberski, U. Thaden, and W. Nejdl. Estimating index staleness under network churn, 2006. Available at http://www.l3s.de/ siberski/reports/index-staleness.pdf.

[166] M. Sintek and S. Decker. TRIPLE - a query, inference, and transformation language for the semantic web. In *Proceedings of the First International Semantic Web Conference*, pages 364–378, Sardinia, Italy, 2002.

[167] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. http://www.mindswap.org/papers/PelletJWS.pdf, submitted for publication to Journal of Web Semantics.

[168] Database language SQL – Part 1: SQL/Framework, 1999. ANSI/ISO/IEC International Standard 9075-1:1999.

[169] T. Stading, P. Maniatis, and M. Baker. Peer-to-peer caching schemes to address flash crowds. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002.

[170] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press New York, NY, USA, 2001.

[171] M. Stonebraker. The design and implementation of distributed INGRES. In *The INGRES papers: anatomy of a relational database system*, pages 187–196. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[172] M. Stonebraker, P. M. Aoki, R. Devine, W. Litwin, and M. A. Olson. Mariposa: A new architecture for distributed data. In *Proceedings of the Tenth International Conference on Data Engineering (ICDE)*, pages 54–65, Houston, Texas, USA, 1994.

[173] subito – Dokumente aus Bibliotheken, 2006. http://www.subito-doc.de/.

[174] H. Suleman and E. A. Fox. A framework for building open digital libraries. *D-Lib Magazine*, 7(12), 2001.

[175] A. N. Swami. Optimization of large join queries: Combining heuristic and combinatorial techniques. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 367–376, Portland, OR, USA, 1989.

[176] Schlagwortnormdatei, 2005. http://www.ddb.de/standardisierung/normdateien/swd.htm.

[177] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, Rome, Italy, 2001.

[178] C. Tempich, S. Staab, and A. Wranik. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *Proceedings of the 13th international conference on World Wide Web*, pages 640–649, New York, NY, USA, 2004.

[179] U. Thaden. *Top-k Retrieval in Peer to Peer Networks*. PhD thesis, Universität Hannover, 2005. http://edok01.tib.uni-hannover.de/edoks/e01dh05/504406213.pdf.

[180] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of Disco. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 449–457, Hong Kong, 1996.

[181] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high-performance peer-to-peer content and resource sharing systems. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, USA, Jan 2003.

[182] P. Valduriez and E. Pacitti. Data management in large-scale P2P systems. In *Proceedings of the 6th International Conference on High Performance Computing for Computational Science (VECPAR 2004)*, pages 104–118, Valencia, Spain, 2004.

[183] H. Van de Sompel and C. Lagoze. The Santa Fe convention of the Open Archives Initiative. *D-Lib Magazine*, 6(2), 2000.

[184] S. Weibel, J. Kunze, and C. Lagoze. Dublin core metadata for resource discovery. Technical report, The Internet Society, 1998. IETF RFC2413, http://rfc.net/rfc2413.html.

[185] W. A. Wiegand and J. Donald G. Davis, editors. *Encyclopedia of Library History*. Garland Publishing, New York & London, 1994.

[186] R. Wilensky. UC Berkeley's digital library project. *Commun. ACM*, 38(4):60, 1995.

[187] I. Witten and D. Bainbridge. *How to build a digital library*. Morgan Kaufman, San Francisco, 2003.

[188] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes*. Morgan Kaufman, Heidelberg, 1999.

[189] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proceeding of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, 2002.

[190] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002.

[191] Information retrieval (Z39.50): Application service definition and protocol specification, 2003. ANSI/NISO Standard Z39.50-2003, http://www.loc.gov/z3950/agency/Z39-50-2003.pdf.