Verarbeitung ortsbezogener Anfragen in lose gekoppelten, föderierten Systemen – Konzeption, Realisierung, Bewertung

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Thomas Schwarz

aus Stuttgart (Deutschland)

Hauptberichter:	Prof. Dr. B. Mitschang
Mitberichter:	Prof. Dr. F. Leymann
Tag der mündlichen Prüfung:	31. Juli 2007

Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware

2007

Vorwort

Diese Arbeit entstand in der Abteilung Anwendersoftware am Institut für Parallele und Verteilte Systeme der Universität Stuttgart. Meine Mitarbeit in der Forschergruppe Nexus und in deren Fortführung im Sonderforschungsbereich 627 „Umgebungsmodelle für mobile, kontextbezogene Systeme“ hat maßgeblich die Themenstellung dieser Arbeit sowie die Umsetzung der Lösungskonzepte beeinflusst.

Besonders möchte ich mich bei Professor Bernhard Mitschang bedanken, der mir während der gesamten Arbeit viel Freiraum zur Umsetzung eigener Ideen gelassen hat, und der mich stets mit konstruktiven Diskussionen und wertvollen Hinweisen unterstützt hat, so dass ich im komplexen Umfeld des Nexus-Projekts diese Promotionsarbeit herausarbeiten konnte.

Ebenso möchte ich mich bei Professor Frank Leymann dafür bedanken, dass er den Mitbericht bei dieser Arbeit übernommen hat. Seine Begeisterung für den Themenkomplex, seine kritischen Anmerkungen und seine Ideen haben zum Gelingen dieser Arbeit bedeutend beigetragen.

Nicht vernachlässigen möchte ich Professor Kurt Rothermel, der als Sprecher des Gesamtprojekts maßgeblich dessen thematische Ausrichtung und die Verwirklichung der Nexus Vision vorangetrieben hat. Ohne seinen Weitblick und sein Engagement wäre diese Arbeit in dieser Form nicht möglich gewesen.

Ich möchte mich ausdrücklich bei den Kolleginnen und Kollegen im Nexus-Projekt und in der Abteilung Anwendersoftware für die herausragende Arbeitsatmosphäre, die konstruktive Zusammenarbeit und das unermüdliche Bestreben, die Nexus Vision gemeinsam umzusetzen, bedanken. Dies ist wirklich außergewöhnlich. Insbesondere gilt mein Dank Daniela Nicklas, Matthias Großmann, Nicola Hönle und Nazario Cipriano aus dem Nexus-Team der Abteilung, Frank Dürr, Martin Bauer, Tobias Drosdol und Christian Becker aus dem Nexus-Team der Nachbarabteilung, Steffen Volz, Mike Eissele, Martin Kada und Darko Klienec aus den Nexus-Teams an anderen Instituten, sowie Christoph Mangold, Mihaly Jakob, Clemens Dorda, Holger Schwarz und Marko Vrhovnik aus der Abteilung Anwendersoftware. Viele der Wissenschaftler im Nexus-Projekt haben durch ihre Anforderungen und Anregungen die thematische Ausrichtung dieser Arbeit mit beeinflusst. Die Implementierung der präsentierten Konzepte wurde durch zahlreiche studentische Arbeiten unterstützt. Stellvertretend möchte ich hierfür Johannes Enge, Markus Iofcea, Shan Liang und Igor Dub danken.

Schließlich möchte ich mich bei meiner Partnerin Silke Wolff und bei meinen Eltern bedanken, die viel Geduld bewiesen haben und mir den Rücken freigehalten haben. Herzlichen Dank für Eure Unterstützung.

Thomas Schwarz Stuttgart, im September 2006

Inhaltsverzeichnis

Abbildungs- und Tabellenverzeichnis

Kapitel 5 Realisierung der Anfrageverarbeitung

Zusammenfassung

Ortsbezogene Anwendungen nutzen die geographische Position des Benutzers, um ihm darauf maßgeschneiderte Informationen anzuzeigen. Sie gelangen an diese Informationen mittels ortsbezogener Anfragen, die sie an einen Datenanbieter stellen. In dieser Arbeit tritt an die Stelle eines einzelnen Datenanbieters nun eine Integrationsmiddleware, welche die räumlichen Daten und die Kontextdaten vieler einzelner Datenanbieter föderiert, so dass daraus ein einziges umfassendes Modell der Umgebung des Benutzers entsteht. Die Verteilung der Daten auf viele Datenanbieter bleibt dabei für die Anwendungen transparent, so dass für die Anwendungen der Eindruck bestehen bleibt, als kämen die Daten von einem einzelnen Datenanbieter. Dadurch können flexibel und effizient verschiedenste ortsbezogene Anwendungen unterstützt werden. Ebenso können Anwendungen von neuen Datenanbietern profitieren ohne dass sie angepasst werden müssen. Die Datenanbieter sind dabei lose gekoppelt, das heißt, sie sind autonom und können sich nach Belieben in das Gesamtsystem ein- und ausklinken. Ein Datenanbieter geht keine Verpflichtung ein, Daten liefern zu müssen. Auch können Datenanbieter ihre eigenen Daten jederzeit aktualisieren oder ergänzen.

Diese Arbeit entstand im Rahmen der Forschergruppe Nexus und des Sonderforschungsbereichs 627 „Umgebungsmodelle für mobile, kontextbezogene Systeme“, wobei sich die vorliegende Arbeit auf die Konzeption und Umsetzung der Integrationsmiddleware des Nexus-Systems konzentriert. Die wichtigste Charakteristik ist dabei die Offenheit des Systems für neue Daten und Datenanbieter.

Die Arbeit untersucht dabei zum einen, wie eine solche Integrationsmiddleware in einem solchen Umfeld prinzipiell funktioniert, und wie dabei die Besonderheiten der Anwendungsdomäne zu deren Vereinfachung ausgenutzt werden können. Zum anderen werden Techniken entwickelt, welche die Charakteristika der Anwendungsdomäne der ortsbezogenen Anwendungen ausnutzen, um die Effizienz der Integrationsmiddleware zu steigern.

Die Basisarchitektur der Integrationsmiddleware setzt sich aus einer Föderationskomponente und einem Verzeichnisdienst zusammen. Die Föderationskomponente hat dabei die selbe Schnittstelle wie die Datenanbieter, wodurch sich Gebietsanfragen recht einfach verarbeiten lassen. Die automatische Koordinatentransformation kann Geodaten, die in unterschiedlichen Koordinatensystemen vorliegen, beispielsweise weil sie von verschiedenen Datenanbietern stammen, automatisch in ein adäquates gemeinsames Koordinatensystem umrechnen, so dass diese gemeinsam verarbeitet und verglichen werden können. Die Verteilung der Daten auf mehrere Anbieter und deren Zusammenführung in der Integrationsmiddleware verursacht das Phänomen der Mehrfachrepräsentationen, weshalb das Datenmodell Mehrfachattribute unterstützen muss. Die dadurch herbeigeführte Komplexitätssteigerung muss entsprechend in der formalen Definition der Anfragesemantik berücksichtigt werden und führt zu vier paarweise dualen Semantikvarianten.

Des Weiteren werden einige Konzepte vorgestellt, um mittels der Charakteristika der Anwendungsdomäne die Effizienz der Anfrageverarbeitung zu steigern. Um die Vollständigkeit und Korrektheit von Anfrageergebnissen, die Mehrfachrepräsentationen enthalten, garantieren zu können, müssen diese zwingend mit speziellen Relationenobjekten verknüpft sein, damit der hierzu entworfene Algorithmus

die Einzelteile bei der Behandlung von Mehrfachrepräsentationen zusammentragen kann. Gegenüber der Garantie freien Verarbeitung von Anfragen können hierbei zusätzliche Interaktionen mit den Datenanbietern notwendig werden.

Der Ortsbezug der Daten und Anfragen wird bei der Verarbeitung föderierter Nachbarschaftsanfragen ausgenutzt. Durch eine iterative Vorgehensweise und eine geschickte Wahl der parallel angefragten Datenanbieter können sowohl die Antwortzeit als auch der Aufwand optimiert werden.

Bei der orts- und typbezogenen Anfrageverarbeitung werden bekannte Indexverfahren geschickt konfiguriert, um eine kombinierte Indexstruktur in den räumlichen Dimensionen und der Typdimension zu erhalten, um so die typischen Anfragen noch effizienter verarbeiten zu können.

Schließlich werden Mechanismen entwickelt, um beliebige domänenspezifische Funktionalitäten in die Anfrageverarbeitung integrieren zu können. Die funktionalitäten können dort zum einen vom direkten Zugriff auf das Umgebungsmodell profitieren. Zum anderen schafft dies die Voraussetzungen, um deren Ergebnisse für andere Anwendungen oder Benutzer wiederverwenden zu können.

Insgesamt werden in dieser Arbeit die Grundprobleme gelöst, wie sich verteilte Umgebungsmodelle lose gekoppelter Datenanbieter föderieren lassen, wie ortsbezogene Anfragen in einem solchen Umfeld verarbeitet werden können, und wie sich der Kontextbezug und weitere Spezifika der Anwendungsdomäne zur Steigerung der Leistung und Effizienz ausnutzen lassen.

Summary

Location-based applications leverage the user's position in order to tailor the presented information to his current location. Those applications retrieve such information by issuing spatial queries to a data provider. In this thesis, we replace the single data provider by an integration middleware that federates the spatial data and the context data of many data providers leading to a single, extensive model of the user's environment. The distribution of the data across several data providers remains transparent for the applications, so that they still perceive the data to originate from a single data provider. The data providers are loosely coupled, meaning that they remain autonomous and that they may enter and leave the federation whenever they like. A data provider has no obligation to supply any data, but it has the opportunity to update or supplement its own data at any time. The core integration concept is a common, global data model, which targets the entire application domain and not just a single application. This allows to support very different location-based applications in a flexible and efficient way, and encourages data reuse by diverse applications. Also, applications may profit from new data providers without having to be modified.

The contents of this thesis has been developed in the context of the research group Nexus and the center of excellence 627 „context models for mobile, context-aware systems". This thesis focuses on the concepts and implementation of the Nexus systems's integration middleware. The system's most important characteristic is its openness for new data and new data providers.

In this thesis, we investigate the architectural basics of such an integration middleware for the above described scenario and stress the simplifications that arise by leveraging the specifics of the application domain. Furthermore, we develop techniques that increase the integration middleware's efficiency by considering the characteristics of the application domain of location-based applications.

The integration middleware's basic architecture comprises a federation component and a discovery service. The federation component features the same interface as the data providers. Hence, the query engine for processing spatial queries can be very simple and forward the queries to the relevant providers obtained from the discovery service. The integration of the returned results can be as simple as concatenating the individual results and merging the data corresponding to the same real world entity. Automatic coordinate transformations transform spatial data with different coordinate systems originating from different data providers into a common coordinate system, which is prerequisite for processing spatial predicates and for applying spatial operators. The distribution of the data across several data providers and the merging of this data in the integration middleware leads to the phenomenon of multiple representations. Therefore, the data model needs to support multi attributes. At the same time, this increases the complexity of data processing, which has to be reflected in the formal definitions of the query semantics and which leads to four pairwise complementary variants of the query semantics.

Furthermore, we present concepts to make query processing more efficient by leveraging the characteristics of the application domain. In order to ensure the completeness and correctness of query results containing multiple representations, those multiple representations need to be linked by special relationship objects. This

allows the proposed query processing algorithm to collect all distributed parts of a multiple representation. Compared to a best effort processing approach that does not guarantee anything, additional interactions with data providers may become necessary.

We exploit the location-awareness of data and queries when processing federated nearest neighbor queries. By using an iterative approach and by cleverly determining the set of data providers addressed in parallel in each iteration we optimize response time and effort at the same time.

We have developed a location and type-aware query processing engine that cleverly configures well known indexing approaches in order to obtain an index structure that combines the spatial and type dimension. This allows to process typical queries more efficiently.

Finally we have developed mechanisms that allow to integrate arbitrary domain-specific functionality into the query engine. Such functionality has the advantage of being able to directly access the context model. Additionally, the integration into the query engine is a prerequisite for reusing intermediate results for other users or applications.

To put it in a nutshell, this thesis addresses the basic problems of federating distributed context models of loosely coupled data providers, of processing location-based queries in such an environment, and of exploiting context information and other specifics of the application domain for increasing performance and efficiency.

In the following, we detail on the major contributions of this thesis.

Architecture

The architecture of the Nexus system [NGS+01, SHG+04, MNG+05] is structured into three tiers: data providers, integration middleware, and applications, which mostly run on mobile devices. As the Nexus system aims at being flexible and open, the integration middleware needs to cope with dynamically changing configurations of data providers. The key integration concept is a common data model [Nick05] that provides the required interoperability between applications and data providers, and that is tailored to the requirements in this open environment. It is accompanied by a specific data exchange format that supports multi attributes and multi types, and by a specific query language that is tailored to the needs of typical context-aware applications. There, simple selection and projection query capabilities suffice.

A data provider stores a local context model which represents a part of the real world as seen from the provider's perspective. Hence, a data provider is quite similar to a web server in the world wide web. For being integrated into the Nexus system, a data provider has to satisfy two requirements. First, it needs to provide a specific query interface, that takes the specific query language as input and that returns the results in a specific data exchange format. This implies that the provider's data is structured according to the common data model. Secondly, the data provider needs to register at the discovery service and provide its service area and a list of the stored object types. This is the prerequisite for being considered in feder-

ated query processing. Depending on the characteristics of the stored data, a data provider has several options for implementing its service [GBH+05].

The integration middleware consists of the federation component and the discovery service. The federation component connects applications and data providers. As it does not store any data itself, it can be easily duplicated in order to support more users. The federation component has the same query interface as a data provider. Thus, a federation component may also act as a data provider that is connected to a superior federation component leading to a hierarchy of federation components [DSB+04]. The federation component employs and executes all the major contributions of this thesis: federated query processing including federated nearest neighbor queries, query semantics dealing with multi attributes and multiple representations, query processing for multiple representations, automatic coordinate transformations, a type and space conscious query engine, and domain-specific query operators.

The discovery service, which is the second part of the integration middleware, keeps track of the currently available data providers. This requires the data providers to register and deregister at the discovery service. The federation component uses the discovery service to determine the relevant data providers for a given query. For this, the discovery service stores a summary of the contents of each data provider. This summary consists of the data provider's service area and of the list of object types of the objects stored there. This is due to the characteristics of typical queries of context-aware applications restricting the qualifying objects at least by type and location. Hence, the discovery service is a spatial directory as well.

The location-based or context-aware applications typically run on mobile devices. They may access the Nexus system in three different ways. First, they may issue spatial queries to the federation component in order to get context information on the user's surroundings. Secondly, they may register events described by predicates at the Event Service in order to receive a notification when the event gets triggered, e.g., when a user enters a building, or when the temperature in a room exceeds a certain threshold. Thirdly, applications may utilize value-added services in order to receive preprocessed context data in special formats, like a road map in the format of an image.

Query Semantics

The semantics of a query is mainly determined by two orthogonal aspects: how to process multi attributes and how to process missing attributes, the latter ones being comparable to NULL values in a database. When processing multi attributes a predicate can be evaluated to true, either if at least one of the attribute's instances satisfies the predicate (Exist-semantics), or if all instances satisfy the predicate (All-semantics). The result of a predicate with no corresponding attribute instances may either be true (Weak-semantics) or false (Strict-semantics). The semantics of a query is always determined by combining both aspects.

We formally define the foundations for processing queries that consist of several booleanly linked predicates. The query engine may either process the predicates individually and combine their results using set operations, or it may process the

entire boolean expression in a single step. This choice provides opportunities for an optimizer. Special attention has to be paid to processing negations, which may lead to inverting the semantics in order to compute them constructively.

Federated Nearest Neighbor Queries

The concept of federated nearest neighbor queries (FNNQ) deals with processing nearest neighbor queries in a federation of loosely coupled data providers storing spatial data. We present the FNNQ algorithm [SIG+04] for processing such queries, describe several variants and compare them using experiments. The algorithm aims at optimizing the number of queries sent to the data providers by dynamically enlarging and shrinking the query area, which determines the set of data providers that are relevant at a certain state of the algorithm. Compared to previous approaches we leverage additional parameterization options: we control the degree of parallelity and we determine the relevant data providers using their service area and a global estimate of the object density. This leads to considerable performance improvements. By adapting the algorithm's parallelity we can trade in response time for resource consumption. Our experiments show that the variant using remote nearest neighbor queries, a global estimate of the object density, and 1+log threads achieves the best overall performance. The approaches from literature may be employed to implement the local processing of nearest neighbor queries at a data provider. Still, the FNNQ algorithm can do without remote nearest neighbor queries as well, and use remote window queries instead. The performance degradation is measurable, but not drastic. To put it all together, our discussion on processing nearest neighbor queries in a federated environment considerably extends the current knowledge on nearest neighbor queries.

Multiple Representations

Multiple representations are objects representing the same real world entity at different data providers. By combining these distributedly stored representations into a single object we can get a more complete picture of the entity. This problem has been addressed previously in the area of data integration, and several generic as well as specific solutions have been proposed for detecting and merging duplicates or for resolving conflicts, see, e.g., [BlNa05]. However, the specific problems pertaining to federated query processing if the data remains at the original data sources have not been investigated so far.

Our analyses show that the processing of multiple representations considerably increases the processing effort, especially if the data used to determine the qualifying objects is distributed across several data providers, and if, at the same time, the result set has to be correct and complete. In the result set, a multiple representation is supposed to appear as a single object, that contains the data of all original representations. Three issues cause the additional processing effort. First, the federation component needs to adapt the semantics of the queries forwarded to the data providers. The data providers return more representations which are possibly discarded in the federation component. Secondly, the federation component may need to fetch special relationship objects and missing representations from the data pro-

viders in two additional steps. Finally, queries in Exist-semantics need to be transformed into a disjunction of single predicates in order to guarantee the result set's completeness. This is quite unfavorable as the Exist-Strict-semantics is the most intuitive and therefore most frequently used semantics. Another requirement for producing complete and correct result sets is the existence of the special relationship objects for all multiple representations. Other duplicate detection methods might not receive all representations if they do not qualify by themselves for the initially forwarded query. The presented algorithm requires additional effort for processing regular objects as well, as the algorithm cannot distinguish them from multiple representations in advance.

We can accelerate the processing if we do not require complete and correct result sets. Then, it is sufficient to execute the algorithm's first step which retrieves only those representations that match the initially forwarded query. No special relationship objects are exploited, and no missing representations are fetched. As a consequence, the result set may contain incomplete objects, duplicates and false positives. However, the false positives are incomplete objects as well, and the false positive's part contained in the result set actually satisfies the query. Missing special relationship objects lead to similar consequences.

Type and Space conscious Main Memory Query Engine

The type and space conscious main memory query engine exploits the characteristics of typical queries, which contain spatial predicates and predicates restricting the type of the returned objects [SGNM06]. Many components of the Nexus system may employ such a query engine. In each usage scenario the query engine is supposed to achieve response times of about ten milliseconds. Both aspects advocate for a main memory approach.

In order to determine the query engine's internal data structure we investigate four approaches ranging from separate indexes on type and geometry over combinations of spatial indexes to a single multidimensional index. In the latter approach we evaluate different strategies for mapping an object type to a value in the type dimension. Our experiments show that the best internal data structure depends on the constraints derived from the usage scenario. Therefore, the query engine needs to be flexible and allow to easily change the internal data structure. Averaged across all usage scenarios, the approach using a spatial index per type yields the best performance. Without a suitable parameterization the initially favoured approach using a single multidimensional index does not perform very well. Still, most previous approaches from literature neglect this parameterization issue. We show that by using the type mapping variants presented in this thesis and by using a proper range for the mapped values the performance can be increased substantially. Yet, the type mapping range needs to be fixed initially and cannot be altered later on without rebuilding the entire index. The best type mapping variant is the one that distributes the mapped values according to the relative frequency of the corresponding type. This variant depends on additional statistics on the frequency of object types. However, even using these parameterization tweaks the approach using a single multidimensional index is in most cases about 10 to 20 percent behind the approach that cleverly combines ordinary spatial indexes.

Automatic Coordinate Transformations

A very important building block for context-aware applications, especially in an open and heterogeneous system landscape, is the capability to process and relate geometries whose coordinates are given in different coordinate systems. The automatic transformation of coordinates into arbitrary coordinate systems [SHGN04] facilitates the usage of existing data, as they do not need to be transformed into a suitable coordinate system in advance. It also facilitates surveilling new data, e.g., by defining a local coordinate system that is specifically tailored to the building whose rooms should be added to the context model. We categorize coordinate systems into local coordinate systems, projected coordinate systems, and geographic coordinate systems. Each coordinate system is represented as a node in the transformation graph. The nodes are connected by transformation rules. The European Petroleum Survey Group [EPSG04] maintains a database describing the most commonly used coordinate systems and the corresponding transformation rules. The automatic transformation is then composed by the transformations along the shortest path in the graph between the nodes corresponding to the source and target coordinate system. Furthermore, each coordinate system is associated with an area of use, which has to be considered when looking for a common coordinate system for jointly processing two geometries. The geometries of arbitrary objects in the context model may now be processed jointly. This is an important foundation ensuring the interoperability of the context data and ensuring the integration middleware's flexibility.

Domain-specific Query Operators

One of the major requirements of the integration middleware of the Nexus system is to integrate domain-specific query operators into query processing. Such operators are supposed to provide frequently used functionality, which applications can build upon and thus need not implement themselves. Additionally, applications may leverage complex and resource consuming functionalities, which would be overcharging a typical mobile device. Processing domain-specific query operators in the integration middleware allows to exploit synergies. The operators may access the context model very quickly and efficiently, allowing them to reduce response times and to better respond to the user's context. Furthermore, the integration middleware may cache the results of such operators and reuse them when processing queries for other users or applications. Domain-specific query operators extend far beyond the processing capabilities of traditional database systems. We investigate two types of operators: value-added services and result set enhancement operators.

Value-added services typically run in the federation component [NGS+01]. However, they are not part of the query processing itself. Value-added services offer their own, proprietary interface, which is specifically tailored to the provided functionality. Value-added services use the federation component's query processing capabilities for accessing the context model. Additionally, they may inquire other data sources as well. Typical value-added services are the navigation service, which computes a route between given start and end points, and the map service, which draws a map with buildings, roads, and other points of interest and returns the map as an image or as vector graphics.

The federation component executes result set enhancement operators as part of the regular query processing [NGS+01]. Such operators process the data fetched from the data providers before it is sent to the applications. Result set enhancement operators come in two flavors: aggregating operators and transforming operators. Aggregating operators accept several result sets as input and return a single, new result set. Each input result set contains the data of one data provider. Aggregating operators typically merge the data of several objects in the input result sets into a new object, which is added to the output result set. An example for an aggregating operator is the conflation operator, which merges the road objects of overlapping road data sets into a single road network.

Transforming operators do only have input result sets, but no output result set. The data is changed in place, so that the input result sets serve as the output result sets at the same time. Transforming operators typically change only specific aspects of an object, so that copying the entire object is an unnecessary overhead. An example for a transforming operator is the generalization operator, which simplifies the floor plans of buildings.

Conclusion

In this thesis, we investigated the processing of spatial queries in loosely coupled federated systems and implemented the results of our investigations in the Nexus system. We designed the Nexus system's architecture to best satisfy the imposed requirements. We analyzed the query processing in the federation component on the one hand focused on the interactions between the components of the Nexus system, and on the other hand focused on the internal processing in the federation component. In the course of this thesis, we solved the basic problems of federating distributed context models of loosely coupled data providers, of processing spatial queries in such an environment, and of exploiting the context-awareness and other specifics of the application domain for increasing efficiency and performance.

The presented concepts addressing the interactions between the components of the Nexus system take care of the data providers' autonomy by using the discovery service for determining the relevant providers. We presented concepts to integrate the data of different providers and to make the data's distribution transparent for the applications. Processing queries is simpler and more efficient by leveraging the specifics of the application domain like the context-awareness of the applications and of their queries. The discovery service exploits this when determining the relevant providers. When processing window queries, all relevant data providers may be inquired in parallel. When processing federated nearest neighbor queries we derive a suitable query area from the query's context. Relationship objects for linking multiple representations may store a position for easier location-based retrieval. Hence, we presented efficient concepts addressing the interaction between components of the Nexus system when processing federated queries.

The concepts addressing the internal processing of queries within the federation component exploit the specifics of the application domain as well. The type and space conscious main memory query engine exploits the characteristics of typical queries that contain spatial predicates and type predicates. Automatic coordinate transformations enable the processing of heterogeneous geodata. Value-added ser-

vices and result set enhancement operators allow to integrate domain knowledge and specific functionalities into query processing. Hence, we thoroughly discussed the internal processing of queries in the federation component and presented concepts to increase performance and efficiency.

The majority of the concepts presented in this thesis have found the acceptance of peer reviewers and have been published at international conferences. The Nexus system's basic architecture and overall concept is described in [NGS+01, SHG+04, MNG+05]. We present federated nearest neighbor queries in [SIG+04]. We describe automatic coordinate transformations in [SHGN04]. We present ideas for the cooperation of federation components in order to improve the management of mobile objects in [DSB+04]. We describe the pros and cons of different implementations for providing a local context model in [GBH+05]. In [HKNS05] we present an extension of the data model for representing meta data. The type and space conscious main memory query engine is explained in [SGNM06].

Most of the concepts presented in this thesis have been implemented in the prototype of the Nexus system, so that applications and other components may now leverage them. The implementation of the federation component was shown at an international conference [NGS03] and at the inspection of the center of excellence 627 in June 2006. The interfaces and data exchange formats are defined in [BDG+04]. Numerous other implementation details are documented in technical reports. Thus, this thesis provides an important foundation for further research conducted by the project partners in the Nexus-project.

1

Einleitung

Ortsbezogene Anwendungen (engl. location based services, LBS) erfreuen sich in den letzten Jahren wachsender Beliebtheit und die Umsetzung in Produkte für den Endverbraucher schreitet unaufhörlich voran. Die Anwendungen nutzen die geographische Position des Benutzers, um ihm darauf maßgeschneiderte Informationen anzuzeigen. Typische Vertreter solcher Anwendungen sind Navigationssysteme oder Touristenführer [AAH+97, CDM+00], um nur die bekanntesten zu nennen. Diese stellen meist Umgebungskarten, Navigationshinweise und Auskünfte über besonders interessante Orte bereit. Diese Anwendungen stellen räumliche Anfragen an eine Datenhaltungskomponente, um an die relevanten Informationen zu gelangen. Dabei spielt der Aufenthaltsort des Benutzers direkt oder indirekt eine zentrale Rolle, wenn beispielsweise der Kartenausschnitt auf den Benutzer zentriert wird oder wenn die nächstgelegenen Hotels gesucht werden. Einen Schritt weiter gehen Anwendungen im Bereich des Pervasive Computing, wo „intelligente“ Alltagsgegenstände miteinander kommunizieren und zusammenarbeiten, um einem Benutzer Dienste und Informationen bereitzustellen. Zusätzlich zu der impliziten Interaktion mit dem Benutzer beziehen solche Anwendungen auch die Umgebung (Wo sind Türen? Welche Ausgabegeräte sind in der Nähe?) und die Tätigkeit des Benutzers (Befindet er sich in einer Sitzung? Kann er gerade gestört werden?), den sogenannten Kontext, in ihre Berechnungen mit ein und passen ihr Verhalten entsprechend an. Typische Vertreter solcher kontextbezogenen Anwendungen sind Gebäudeinformationssysteme und intelligente Umgebungen [BaKi01, CKW01, KOA+99]. Dennoch stellt auch hier die Position des Benutzers einen sehr wichtigen, wenn nicht gar den wichtigsten, Teil des Kontexts dar. Zusammen mit anderen räumlichen Daten wie beispielsweise digitalen Kartendaten, Gebäudemodellen oder markanten Punkten ergeben die Kontextdaten ein Umgebungsmodell, das die Umgebung eines Benutzers digital nachbildet, damit diese von Anwendungen verarbeitet werden kann.

1.1 Status Quo

Bisher wurden schon eine Reihe von Systemen und Infrastrukturen vorgeschlagen, die solche Anwendungen unterstützen und deren Entwicklung vereinfachen. Diese zielen jedoch überwiegend auf sehr eng abgesteckte Anwendungsgebiete wie die

Verarbeitung von Kontextdaten, die von Sensoren stammen [SDA99], ein Touristeninformationssystem [CDM+00] oder die Verwaltung von Positionsinformationen [STM00]. Zudem sind die Systeme recht unflexibel wenn neue Dienste hinzugefügt werden sollen oder wenn neue Umgebungsinformationen wie Kartendaten oder Informationen über Sehenswürdigkeiten verfügbar werden. Selbst wenn eine solche Erweiterung für eine bestimmte Anwendung funktioniert, dann heißt dies noch lange nicht, dass auch andere Anwendungen davon profitieren können. Auch ist eine Interaktion zwischen verschiedenen Anwendungen, z. B. in Form eines Austauschs von Identitäten, Orten oder Zuständen, nicht möglich, wenn nicht beide Anwendungen ihre Kontextdaten in übereinstimmender Weise repräsentieren.

Für den Bereich der ortsbezogenen Anwendungen existieren schon einige kommerzielle Lösungen, die von Betreibern von Mobilfunkportalen oder auch in Fahrzeugnavigationssystemen genutzt werden. Diese Lösungen basieren auf räumlichen Daten, die typischerweise von einem Datenlieferant eingekauft werden und die für die beabsichtigte Anwendung speziell aufbereitet werden. Viele Daten werden von Hand erfasst, angepasst, aufbereitet oder aktualisiert, was diese Daten teuer macht. Als Folge davon wird der Zugriff auf die Daten stark eingeschränkt. Zahlende Kunden können die Daten ihres Datenanbieters nutzen, aber auch eben nur diese. Meist enthalten solche Systeme keine umfassenden Informationen, wie z. B. alle Hotels in einem Stadtteil, sondern nur Informationen von Vertragspartnern. Durch das zentralisierte Betreiberkonzept gestaltet sich die Aktualisierung des Datenbestands und die Erweiterung um neue Informationen und insbesondere um neue Informationskategorien recht schleppend.

1.2 Zielsetzung

Demgegenüber entsteht im Nexus-Projekt [NGS+01] eine offene Plattform, die ein umfassendes Umgebungsmodell verwaltet. Das Nexus-System föderiert die räumlichen Daten und die Kontextdaten vieler einzelner Datenanbieter, um flexibel und effizient verschiedenste ortsbezogene Anwendungen unterstützen zu können. Die Datenanbieter sind dabei lose gekoppelt, das heißt, sie sind autonom und können sich nach Belieben in das Nexus-System ein- und ausklinken. Ein Datenanbieter geht keine Verpflichtung ein, Daten liefern zu müssen. Auch können Datenanbieter ihre eigenen Daten jederzeit aktualisieren oder ergänzen. Die Interoperabilität zwischen Anwendungen und Datenanbietern wird durch ein global festgelegtes, domänenspezifisches Schema, sowie durch darauf aufbauende Schnittstellendefinitionen und Datenaustauschformate gewährleistet. Das System kann dabei leicht um zusätzliche Datenanbieter und Informationskategorien erweitert werden. Die Grundidee der Föderation besteht darin, dass eine Integrationsmiddleware die Verteilung der Daten auf viele Datenanbieter transparent für die Anwendungen macht, so dass für diese der Eindruck eines einzigen umfassenden Umgebungsmodells entsteht, das an einer zentralen Stelle verwaltet wird, obwohl tatsächlich die Daten bei den Anbietern verbleiben. Anwendungen können von neuen Datenanbietern profitieren ohne dass sie angepasst werden müssen. Sie sind weniger abhängig von einzelnen Datenanbietern und funktionieren weiter, wenn einer verschwinden sollte.

Derzeit gibt es keine Technologie, um eine umfassende Integration der Daten beliebig vieler im voraus unbekannter Datenanbieter in generischer Weise zu erreichen.

Der in dieser Arbeit verfolgte Lösungsansatz zielt auf die Erstellung einer spezifischen Integrationsmiddleware, die auf eine bestimmte Anwendungsdomäne (wie z. B. Biologie, e-business oder wie in dieser Arbeit ortsbezogene Anwendungen) zugeschnitten ist. Anwendungen einer Domäne strukturieren und verarbeiten ihre Daten auf sehr ähnliche Weise, so dass eine domänenspezifische Integrationsmiddleware diese Gemeinsamkeiten ausnutzen kann. Durch sorgfältig abgewogene Einschränkungen wie die eines globalen Schemas und einer vereinfachten Anfragesprache wird es möglich beliebige Anwendungen mit beliebigen Datenanbietern über ein gemeinsames Datenschema, welches in [DiFe01] auch als „upper ontology“ bezeichnet wird, zu koppeln. Die Integrationsmiddleware kann domänenspezifisches Wissen nutzen, um eine semantische Datenintegration und weitergehende Funktionalitäten anbieten zu können: Erkennung und Verschmelzung von Duplikaten beziehungsweise Mehrfachrepräsentationen, oder die Generalisierung, Aggregation oder anderweitige Aufbereitung von Datenobjekten. Komplexe Funktionalitäten, die sonst von einer Anwendung selbst umgesetzt werden müssten, können in die Integrationsmiddleware verlagert werden. Damit kann die Anwendungsentwicklung komfortabler gestaltet werden, die Funktionalitäten können effizienter ausgeführt werden und die Ergebnisse können für andere Benutzer und andere Anwendungen wiederverwendet werden. Daten werden nicht mehr nur von einer speziellen Anwendung genutzt, sondern gleichzeitig von vielen verschiedenen Anwendungen. Umgekehrt ist eine Anwendung nicht mehr auf einen einzigen Datenanbieter beschränkt, sondern kann die Daten vieler verschiedener Anbieter nutzen, die sich gegenseitig ergänzen.

Dies geht auch weit über die Fähigkeiten von Suchmaschinen im Web, wo die Inhalte für einen menschlichen Leser optimiert sind, und des Semantic Web Ansatzes, wo Webseiten mit zusätzlichen semantischen Informationen angereichert sind, hinaus. Nach der Vision des Semantic Web wandelt sich das Web der Seiten hin zu einem Web der Objekte. In dieser Vision wird die automatische Verarbeitung von Informationen im Web angestrebt. Nicht behandelt wird das Problem, dass die Informationen über viele Datenanbieter verstreut sind. Das Auffinden und Verarbeiten der richtigen Informationen bleibt eine mühevolle Aufgabe, die typischerweise von automatischen Softwareagenten durchgeführt wird. Dennoch muss ein solcher Softwareagent wissen, wo er suchen soll, und er muss die verschiedenen Schemata der Datenanbieter kennen, was den Automatismus und die Interoperabilität stark einschränkt.

1.3 Beitrag

In dieser Arbeit wird die Konzeption der Integrationsmiddleware des Nexus-Systems vorgestellt, siehe Abbildung 1. Verschiedene Aspekte der Anfrageverarbeitung werden detailliert untersucht und konkurrierende Ansätze, wo vorhanden, experimentell verglichen. Dabei werden insbesondere die folgenden Grundprobleme betrachtet:

- Wie lassen sich die verteilten Umgebungsmodelle der lose gekoppelten Datenanbieter so zusammenführen bzw. föderieren, dass für die Anwendung der Eindruck entsteht, dass alle Daten zentral verwaltet werden?

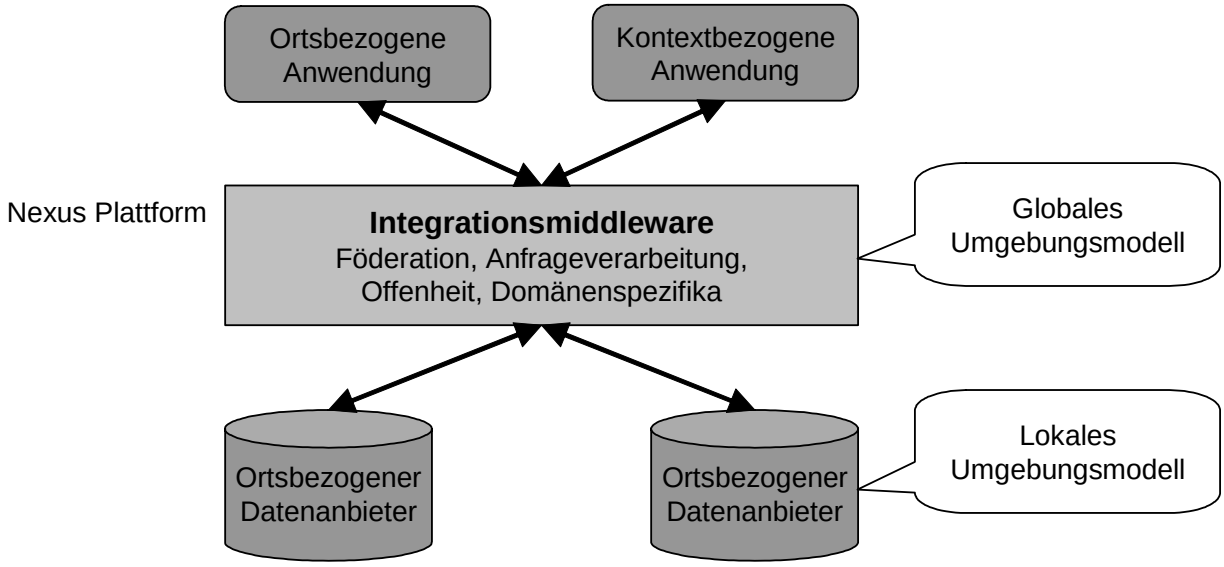


Abbildung 1: Integrationsmiddleware des Nexus-Systems

- Wie baut man eine ortsbezogene Anfrageverarbeitungskomponente in lose gekoppelten, föderierten Systemen?
- Wie lässt sich die Leistung und Effizienz einer Föderationskomponente durch Ausnutzen von Spezifika der Anwendungsdomäne wie beispielsweise des Kontextbezugs steigern?

Diese Arbeit entstand im Rahmen der Forschergruppe Nexus und des Sonderforschungsbereichs 627 „Umgebungsmodelle für mobile, kontextbezogene Systeme“. Während sich die vorliegende Arbeit auf die Konzeption und Umsetzung der Integrationsmiddleware konzentriert, werden im Nexus-Projekt zahlreiche weitere Themen bearbeitet, wie beispielsweise der Entwurf des Umgebungsmodells [Nick05], die effiziente Verwaltung stationärer Objekte [GBH+05] sowie mobiler Objekte [LeRo02], der ereignisbasierte Zugriff auf das Umgebungsmodell [BaRo04], die Entdeckung und Verarbeitung von Mehrfachrepräsentationen [VoWa04] oder die Generalisierung von Gebäudegrundrissen [KaFe06], um nur einige zu nennen. Viele der Wissenschaftler im Projekt haben durch ihre Anforderungen, Anregungen und Vorschläge zur Integrationsmiddleware beigetragen. Auch deren Implementierung und Einbettung im Nexus-System wäre ohne die Zusammenarbeit im Gesamtprojekt und die Hilfe zahlreicher studentischer Arbeiten nicht möglich gewesen.

1.4 Übersicht

Die vorliegende Arbeit ist wie folgt strukturiert. In Kapitel 2 werden die Randbedingungen vorgestellt, die sich aus dem Nexus-Projekt ergeben. Dies sind insbesondere die Anforderungen an die Föderationskomponente sowie die Eigenschaften der zu verarbeitenden Daten und der diese verwaltenden Datenanbieter. Der grundlegende Föderationsansatz wird in Kapitel 3 vorgestellt. Hier wird die Architektur des Nexus-Systems beleuchtet, sowie das Datenaustauschformat und die Anfragesprache vorgestellt. Ebenso werden Aufgaben und Funktionsweise des Verzeichnisdienstes sowie alternative Architekturansätze diskutiert. In Kapitel 4 wird das Konzept zur Verarbeitung von Anfragen in der Föderationskomponente

erstellt. Dabei wird insbesondere der Teil der Anfrageverarbeitung erörtert, der die Interaktion mehrerer Komponenten im Gesamtsystem erfordert. Nachdem die Semantik von Anfragen formal definiert und die grundsätzliche Vorgehensweise zur Verarbeitung von Gebietsanfragen vorgestellt wurde, werden verschiedene Ansätze zur Verarbeitung von föderierten Nachbarschaftsanfragen miteinander verglichen. Des Weiteren wird erörtert, welche Auswirkungen die Präsenz von Mehrfachrepräsentationen im Datenbestand auf die Anfrageverarbeitung hat, und wie deren Funktionsweise angepasst werden muss, um eine möglichst vollständige und korrekte Ergebnismenge zu erhalten. Schließlich wird die Kooperation benachbarter Föderationskomponenten als weitere Maßnahme zur Steigerung der Effizienz der Anfrageverarbeitung vorgestellt. Das folgende Kapitel 5 beleuchtet die Realisierung der Anfrageverarbeitung innerhalb einer Föderationskomponente. Hierfür wird zunächst beschrieben, wie die Daten des Umgebungsmodells in geeigneter Weise im Hauptspeicher repräsentiert werden können und wie die charakteristischen Eigenschaften der Anfragen kontextbezogener Anwendungen ausgenutzt werden können, um die Effizienz der Anfrageverarbeitung zu steigern. Solche Anfragen enthalten häufig Einschränkungen des Orts und des Typs der gesuchten Objekte. Anschließend wird aufgezeigt, wie räumliche Daten, die in verschiedenen Koordinatensystemen vorliegen, trotzdem gemeinsam verarbeitet werden können, was essentiell für die Interoperabilität zwischen heterogenen Datenanbietern und Anwendungen ist. Schließlich wird gezeigt, wie verschiedene domänenspezifische Funktionalitäten in die Integrationsmiddleware verlagert werden können, um dort als Teil der Anfrageverarbeitung von vielen verschiedenen Anwendungen genutzt werden zu können. Als Abschluss des Kapitels werden verschiedene Caching-Ansätze zum Zwischenspeichern von Ergebnismengen in der Föderationskomponente erörtert. Eine Zusammenfassung und ein Ausblick in Kapitel 6 beschließen diese Arbeit. In jedem Kapitel werden die verwandten Arbeiten jeweils direkt bei dem zughörigen Verfahren oder Algorithmus diskutiert.

2

Randbedingungen

Dieses Kapitel beschreibt die Randbedingungen, die an eine föderierte Anfrageverarbeitung im Umfeld des Nexus-Systems gestellt werden. Diese Randbedingungen setzen sich aus drei Teilen zusammen. Zunächst werden die Idee des Nexus-Systems und mögliche darauf aufbauende Anwendungen in Kapitel 2.1 vorgestellt. Anschließend werden die aus der Nexus-Idee abgeleiteten Anforderungen an eine Föderationskomponente in Kapitel 2.2 dargelegt. Diese geben die bereitzustellenden Funktionalitäten und die zu lösenden Probleme vor. Schließlich wird in Kapitel 2.3 das Datenmodell des Umgebungsmodells erläutert. Dieses legt den Aufbau von Objekten fest und bestimmt weitere Charakteristika wie beispielsweise den Aufbau von Identifikatoren oder die Möglichkeit für mehrere Typen und mehrfache Attributinstanzen.

2.1 Die Nexus-Idee

Das Ziel der Nexus-Plattform [HKL+99] besteht darin, alle denkbaren ortsbezogenen und, allgemeiner noch, kontextbezogenen Anwendungen in generischer Weise zu unterstützen. Dies betrifft zum einen die Entwicklung der Anwendungen und zum anderen deren Ausführung. Unzählige Anwendungen, die bisher nur als Insellösungen existiert haben, wie beispielsweise Fahrzeugnavigationssysteme oder Museumsführer, sollen nun in einer einzigen offenen Plattform zusammengeführt werden. Die Plattform soll es den Anwendungen ermöglichen, sowohl allgemeine als auch anwendungsspezifische Daten abzurufen, um so den Kontext des Benutzers zu ermitteln und entsprechend darauf reagieren zu können. Ebenso sollen verschiedene Anwendungen miteinander kommunizieren und zusammenarbeiten können. Die Nexus-Plattform ist als eine Middleware konzipiert, die Produzenten und Konsumenten von ortsbezogenen Informationen zusammenbringt.

Die Grundidee zur Integration der verschiedenen Datenanbieter und Anwendungen besteht darin, dass sie alle eine gemeinsame Sprache sprechen. Diese Aufgabe erfüllt das globale Umgebungsmodell, das von der Nexus-Plattform bereitgestellt wird, siehe Abbildung 2. Dieses Umgebungsmodell beschreibt das Umfeld des Benutzers. Es besteht zum einen aus digitalen Repräsentationen von Dingen der realen Welt (reale Objekte) und zum anderen aus virtuellen Objekten, die typischer-

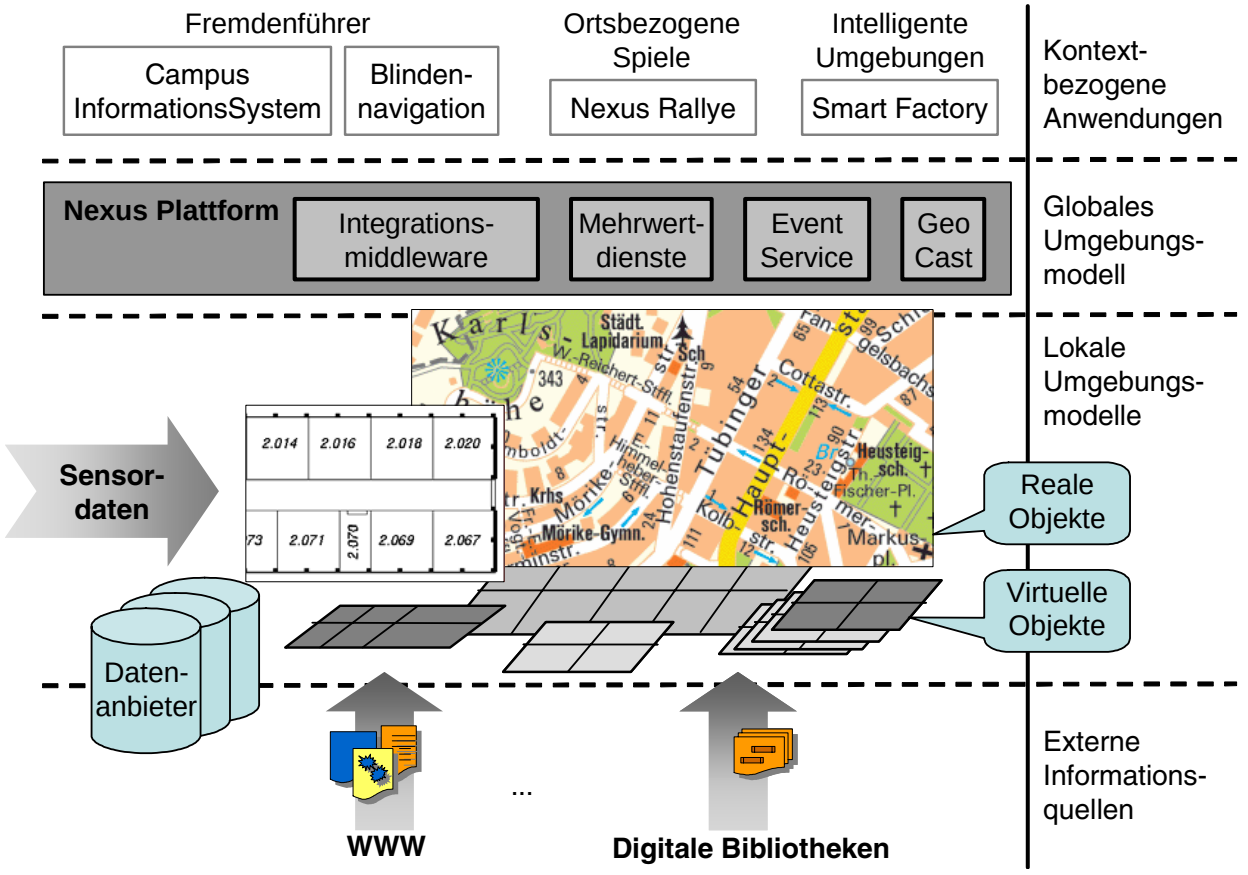


Abbildung 2: Die Nexus-Idee

weise Verweise auf externe Informationsquellen mit einem bestimmten Ort verknüpfen. Die Nexus-Plattform ermöglicht, dass viele verschiedene Anbieter Daten bereitstellen können, ohne dass eine Anwendung diese Verteilung bemerkt. Datenanbieter müssen jetzt keinen umfassenden Datensatz mehr bereitstellen, sondern können sich darauf beschränken, einen kleinen Beitrag zum globalen Umgebungsmodell zu leisten, der durch die Beiträge der anderen Datenanbieter ergänzt wird. Jeder Datenanbieter liefert die Daten, für die er sich zuständig fühlt, und die er damit auch aktuell hält. Diese Offenheit ist vergleichbar mit dem World Wide Web, wo auch jedermann Informationen anbieten kann, indem er einen eigenen Web Server betreibt, oder indem er seine Web Seiten bei einem Web Hoster einstellt. Die Integrationsmiddleware der Nexus-Plattform sorgt dafür, dass das aus vielen lokalen Umgebungsmodellen zusammengesetzte globale Umgebungsmodell nach außen hin wie aus einem Guss aussieht. Die Verwendung der Daten durch viele verschiedene Anwendungen hat einen weiteren Vorteil. Gemeinsam genutzte Daten müssen nur einmal erhoben werden und können dann oft wiederverwendet werden. Der Aufwand für das Erheben der Daten wird durch eine breitere Nutzung belohnt. Änderungen in der realen Welt können mit Hilfe von Sensoren in das Umgebungsmodell übertragen werden. Im Moment ist dies auf simple Messwerte, wie z. B. der Raumtemperatur oder des Öffnungszustands einer Tür, beschränkt. In Zukunft soll jedoch auch die Erkennung komplexer Situationen, wie z. B. einer Sitzung, als virtuelle Sensoren in das Umgebungsmodell integriert werden [KKN+05].

Zur umfassenden Unterstützung der kontextbezogenen Anwendungen gehören eine Reihe von Funktionalitäten, die von der Nexus-Plattform bereitgestellt werden.

Auf die Daten des Umgebungsmodells kann in verschiedener Weise zugegriffen werden. Eine Anwendung kann Daten direkt mittels einer Anfrage anfordern (pull-Modus), sie kann sich über das Eintreten eines bestimmten Zustands benachrichtigen lassen (push-Modus) oder sie kann einen Mehrwertdienst nutzen, der die Daten in aufbereiter Form, beispielsweise als Pixelbild, liefert. Der Kern dieser Arbeit beschäftigt sich mit der Umsetzung der ersten Zugriffsmethode. Mehrwertdienste werden in Kapitel 5.4.1 angesprochen. Die zweite Zugriffsmethode wird im Nexus-System durch den Event Service bereitgestellt, welcher das Thema einer eigenen Promotion [BaRo04] darstellt. Des Weiteren bietet die Nexus-Plattform mit GeoCast einen Dienst, der es erlaubt, Nachrichten an Benutzer zu schicken, die sich in einem gegebenen Gebiet aufhalten. Für die Anbindung der mobilen Endgeräte an die Nexus-Plattform werden Verfahren entwickelt, die einerseits einen nahtlosen Wechsel der genutzen Mobilfunktechnologie und andererseits eine Anpassung der Anwendung je nach zur Verfügung stehender Bandbreite erlauben.

Eine Reihe von Anwendungen im Nexus-Projekt bauen direkt auf den hier entwickelten Konzepten und deren prototypischer Umsetzung auf. Das CampusInformationsSystem [NGS03] erleichtert Besuchern und Studierenden die Navigation auf dem Campus der Universität Stuttgart. Die NexusRallye [NHMM04] lotst Studienanfänger in einer Art elektronischer Schnitzeljagd über den Campus und gibt ihnen in spielerischer Weise eine erste Einführung. Hierbei kann die Erfüllung einzelner Aufgaben automatisch erfasst werden. Ein System zur Unterstützung sensorisch eingeschränkter Menschen kann mittels des Umgebungsmodells einem Benutzer mitteilen, auf welchen Gegenstand er gerade zeigt, ob er sich auf eine Kante oder Treppe zubewegt, oder ob eine Tür geöffnet oder geschlossen ist [HDT04]. In der SmartFactory [BJS04] werden Produktionsprozesse flexibilisiert, indem beispielsweise Werkstücke mit Hilfe des Umgebungsmodells die nächstgelegene verfügbare Maschine für den anstehenden Verarbeitungsschritt ermitteln.

Weitere in der Literatur betrachtete Anwendungen wie Touristenführer, Museumsführer, Fahrzeugnavigationssysteme oder ortsbezogene Spiele sowie intelligente Räume, intelligente Häuser oder allgemein intelligente Umgebungen sind ebenfalls als Nutzer der Nexus-Plattform denkbar. Eine umfassende Analyse der potentiellen Anwendungen und deren Anforderungen an das Umgebungsmodell erfolgt in [Nick05].

2.2 Anforderungen an die Föderationskomponente

Aus der Idee der Nexus-Plattform folgen die folgenden Anforderungen an die zugehörige Integrationsmiddleware beziehungsweise an deren Föderationskomponente.

- **Autonomie**: Die Datenanbieter sind alle selbständig. Sie können kommen und gehen nach Belieben und sie haben keine Verpflichtungen gegenüber der Föderationskomponente. Insbesondere kann die Föderationskomponente keine Anpassungen an einem Datenanbieter vornehmen, um diesen besser integrieren zu können.
  - Die Anforderung wird erfüllt durch den Verzeichnisdienst in Kapitel 3.4 und dessen Nutzung in den Verfahren in Kapitel 4. Die einzige Einschränkung

der Autonomie besteht darin, dass sich die Datenanbieter beim Verzeichnisdienst mit ihrem Dienstgebiet und den Typen der verwalteten Objekte anmelden müssen.

- **Integration**: Die Föderationskomponente soll den Anwendungen eine integrierte Sicht auf die verteilten Daten der verschiedenen Datenanbieter bereitstellen. Das globale Umgebungsmodell soll wie aus einem Guss erscheinen. Insbesondere sollen Duplikate beziehungsweise Mehrfachrepräsentationen vermieden werden.
  - Die in den Kapiteln 4.2 bis 4.4 vorgestellten Verfahren sammeln die Daten von allen relevanten Anbietern ein, unterschiedliche Koordinatensysteme der Geodaten werden mit dem Verfahren aus Kapitel 5.3 angepasst, und Mehrfachrepräsentationen werden mittels eines in Kapitel 5.4.2 beschriebenen Ergebnisaufbereitungsoperators verschmolzen.
- **Transparenz**: Anwendungen sollen sich nicht um die verteilte Speicherung der Daten kümmern müssen, sondern sollen diese gar nicht bemerken.
  - Nach der Nomenklatur aus [BKLW99] wird diese Anforderung wie folgt erfüllt. Das globale Schema des Umgebungsmodells (Kapitel 2.3) sorgt für Schematransparenz. Die Architektur des Gesamtsystems (Kapitel 3.1) sowie die verwendeten Schnittstellensprachen (Kapitel 3.2 und 3.3) sorgen für Heterogenitätstransparenz. Auf einer anderen Ebene sorgen die Hauptspeicherrepräsentationen von Anfragen und Umgebungsmodelldaten (Kapitel 5.1) ebenfalls für Heterogenitätstransparenz. Die deklarative Anfragesprache (Kapitel 3.3) und der Verzeichnisdienst (Kapitel 3.4) sowie das Konzept der Objektidentifikatoren (Kapitel 2.3) sorgen für Orts- und Namenstransparenz. Die Anfrageverarbeitungsverfahren aus Kapitel 4 sowie die Ergebnisaufbereitungsoperatoren aus Kapitel 5.4.2 sorgen für Verteilungstransparenz.

Die drei Anforderungen Autonomie, Integration und Transparenz sind zentral in dieser Arbeit. Föderationsansatz, Föderierte Anfrageverarbeitung und Anfrageverarbeitung innerhalb der Föderation erfüllen im Zusammenspiel diese Anforderungen.

- **Anfragetypen**: Anwendungen möchten entweder alle Objekte mit bestimmten Eigenschaften haben, die sich in einem gegebenen Gebiet, beispielsweise in einem Kartenausschnitt, befinden. Diese Informationsanforderung wird Gebietsanfrage genannt. Alternativ möchten Anwendungen die Größe der Ergebnismenge beschränken, und als Ergebnis diejenigen Objekte mit bestimmten Eigenschaften erhalten, die den geringsten Abstand zu einem gegebenen Referenzpunkt aufweisen. Beispielsweise möchte eine Anwendung die drei nächstgelegenen Tankstellen in einer Auswahlliste darstellen. Diese Informationsanforderung wird Nachbarschaftsanfrage genannt.
  - Entsprechende Verfahren zur Verarbeitung der beiden Anfragetypen werden in den Kapiteln 4.2 und 4.3 behandelt. Die Datenanbieter ihrerseits müssen mindestens Gebietsanfragen verarbeiten können. Nachbarschaftsanfragen sind für Datenanbieter optional aber erwünscht, da dann die von Anwendungen gestellten Nachbarschaftsanfragen robuster und effizienter verarbeitet

werden können, siehe Kapitel 4.3. Die Semantik von Anfragen wird in Kapitel 4.1 formal definiert.

- **Informationsmaximierung**: Die Föderationskomponente soll mit allen möglichen Kombinationen von Datenanbietern arbeiten können, und soll jeweils das bestmögliche mit den aktuell verfügbaren Datenanbietern erzielbare Ergebnis zurückliefern.
  - Die in Kapitel 4 diskutierten Verfahren sind entsprechend ausgelegt.
- **Datenaufbereitung**: Die Föderationskomponente soll zusätzliche domänenspezifische Funktionalitäten zur Aufbereitung der Umgebungsmodelldaten bereitstellen oder dynamisch ausführen können, um so zum einen die Anwendungen auf ressourcenschwachen mobilen Endgeräten zu entlasten, und um zum anderen Synergien erzielen zu können, indem die Ergebnisse der Berechnungen von anderen Benutzern oder anderen Anwendungen mitverwendet werden können. Die aufbereiteten Daten werden im selben Format wie nicht aufbereitete Daten an die Anwendung weitergeleitet und sind im besten Fall von dieser nicht zu unterscheiden.
  - Die in Kapitel 5.4.2 vorgestellten Ergebnisaufbereitungsoperatoren ermöglichen die Integration der gewünschten Funktionalität in die Anfrageverarbeitung. Derzeit können sich überlappende Straßendatensätze verschmolzen werden sowie Gebäudegrundrisse in ihrem Detailgrad angepasst werden.
- **Mehrwertdienste**: Ähnlich wie bei der Datenaufbereitung sollen häufig und von vielen Anwendungen benötigte Dienste in die Föderationskomponente verlagert werden können. Mehrwertdienste dürfen jedoch proprietäre Schnittstellen haben und ihre Ergebnisse in eigenen Formaten zurückliefern.
  - Kapitel 5.4.1 beschreibt die Integration solcher Mehrwertdienste in die Gesamtarchitektur und in die Anfrageverarbeitung. Hierunter fallen unter anderem die Berechnung von Navigationsrouten sowie das Zeichen von Karten.

Neben den bisher genannten funktionalen Anforderungen bestehen auch die folgenden nicht-funktionalen Anforderungen:

- **Effizienz**: Anfragen sollen möglichst effizient verarbeitet werden. Bei der Verarbeitung sollen domänenspezifische Besonderheiten wie beispielsweise der Ortsbezug für Optimierungen ausgenutzt werden.
  - Insbesondere die Verarbeitung föderierter Nachbarschaftsanfragen (Kapitel 4.3), die orts- und typbezogene Anfrageverarbeitung im Hauptspeicher (Kapitel 5.2) sowie die in den Kapiteln 4.5 und 5.5 vorgestellten weiteren Maßnahmen nutzen den Ortsbezug der Daten und der Anfragen für Effizienzsteigerungen. Teilweise wird auch die charakteristische Eigenschaft der Anfragen ausgenutzt, die zusätzlich zum Ort häufig auch den Typ der gesuchten Objekte festlegen.
- **Skalierbarkeit**: Die Verfahren sollen so konzipiert werden, dass sie viele Datenanbieter und viele gleichzeitige Benutzer verkraften können.
  - Die Basisarchitektur ist so ausgelegt, dass Föderationskomponenten beliebig dupliziert werden können, um mehr Benutzer versorgen zu können. Zudem

wurde die Effizienz einiger Verfahren unter Ausnutzung der Besonderheiten der Anwendungsdomäne gesteigert, so dass eine einzelne Föderationskomponente mehr Benutzer bedienen kann.

2.3 Datenmodell

Das Datenmodell und das Schema des Umgebungsmodells wurden festgelegt, nachdem in [Mess01] eine umfassende Analyse der Anforderungen verschiedener ortsbezogener Anwendungen durchgeführt wurde. Die Details hierzu können [NiMi04, Nick05] entnommen werden. Dort wird das Umgebungsmodell auch Augmented World Model (AWM) genannt. In der Terminologie der MOF Metadata Architecture [OMG05] entspricht das Datenmodell dem Meta-Modell, das Schema dem Modell und das Umgebungsmodell der Information. Im Folgenden werden die wesentlichen Aspekte kurz zusammengefasst und anhand des Beispiels in Abbildung 3 erläutert.

Abbildung 3: Beispiel eines Objekts im Umgebungsmodell

Die Daten des Umgebungsmodells sind in Objekte strukturiert. Jedes Ding der realen Welt wird als ein eigenes Objekt repräsentiert, wie beispielsweise das Linden-Museum in Stuttgart. Natürlich kann ein Objekt wiederum andere Objekte enthalten. Ein Haus bzw. Museum besteht aus mehreren Räumen, in einem Raum stehen Möbel. Für eine tiefergehende Diskussion solcher Details sei auf [Mess01, NiMi04, Nick05] verwiesen. Im Wesentlichen bestehen Objekte aus einem Typ, einem Identifikator und einer Reihe von Attributen, wobei bedingt durch die Anwendungsdomäne jedes Objekt mindestens ein Positionsattribut (pos) und ein Grundrissattribut (extent) enthält. Das Positionsattribut wird für die einfache Berechnung von Abständen und für Nachbarschaftsanfragen benötigt, während das Grundrissattribut für Gebietsanfragen („Ist das Objekt im Kartenausschnitt zu sehen?“) und für die Darstellung eines Objekts in einer Karte benötigt wird.

Der Typ eines Objekts entstammt dem Schema des Umgebungsmodells. Der Typ legt fest, welche Attribute erlaubt sind. Hierbei gibt es erforderliche Attribute, die jedes Objekt dieses Typs besitzen muss, sowie optionale Attribute. Letztere haben den Sinn, dass sie für gewisse Aspekte eines Objekts schon den Namen und Datentyp des zugehörigen Attributs vorgeben, so dass Datenanbieter, wenn sie Informationen zu diesem Aspekt haben, diese in gleicher Weise repräsentieren, so dass die Interoperabilität gewährleistet ist. Das Schema legt zu jedem Attribut ebenfalls den Datentyp dieses Attributs fest. Mögliche Datentypen sind Identifikator, Zeichenkette, ganze Zahl, rationale Zahl, Wahrheitswert, Zeitpunkt und Zeitabschnitt sowie Punkt und Geometrie. Beispielsweise hat das Positionsattribut den Datentyp Punkt, das Grundrissattribut den Datentyp Geometrie und der Name eines Objekts den Datentyp Zeichenkette.

Durch die Integration der Daten mehrerer Anbieter kann es vorkommen, dass die Daten zum selben Objekt, welches dann eine Mehrfachrepräsentation ist, sich in ihrem Wert unterscheiden. Prinzipiell könnte die Integrationsmiddleware solche Konflikte auflösen und aus widersprüchlichen Werten einen zusammengefassten Wert ermitteln. Gemäß der Anforderung der Informationsmaximierung wird hier jedoch der Ansatz verfolgt, keine Informationen wegzuwerfen und stattdessen der Anwendung diese Entscheidung zu ermöglichen. Beispielsweise könnte eine Straße nach der Integration zwei Namen besitzen, B14 und Hauptstätter Straße. Während der erste Name für einen Autofahrer relevant ist, der schnell die Stadt durchqueren möchte, ist der zweite Name interessant, wenn ein bestimmtes Haus mit einer gegebenen Hausnummer in dieser Straße gesucht wird. Um dies zu ermöglichen erlaubt das Datenmodell Mehrfachattribute und Mehrfachtypen. Ein Objekt kann vom jedem Attribut, das sein Objekttyp definiert, beliebig viele Instanzen enthalten. Beispielsweise enthält das Museum in Abbildung 3 zwei Ausstellungsattribute. In gleicher Weise können sich die Datenanbieter auch bei der Festlegung des Objekttyps unterscheiden, so dass das Objekt mehrere Typen haben kann. Als Attribute dürfen alle bei diesen Objekttypen definierten Attribute auftreten. Ein Datenanbieter könnte das Museum aus dem Beispiel als Konzertsaal in seinem Bestand führen, weil dort hin und wieder auch musikalische Darbietungen stattfinden. Um dies zu unterstützen wird der Typ eines Objekts wie ein weiteres Attribut aufgefasst, das ebenfalls mehrere Instanzen haben kann. Siehe hierzu auch [SHG+04].

Jedes Objekt besitzt einen Identifikator, der im Nexus-Umfeld auch NOL bzw. Nexus Object Locator genannt wird. Dieser besteht im wesentlichen aus drei Teilen, dem Servernamen, der Datensatz-ID und der Objekt-ID. Vergleiche hierzu auch den NOL des Museums in Abbildung 3. Der Servername bezeichnet den per DNS auflösbaren Namen des Servers des Datenanbieters, der das Objekt speichert. Die Datensatz-ID bezeichnet eine logische Einheit (Datensatz) bei einem Datenanbieter. Dies erlaubt einem Datenanbieter seine Daten zu gruppieren, oder auch Daten im Auftrag verschiedener Kunden anzubieten. Die Objekt-ID ist der einzige eindeutige Bezeichner eines Objekts (der realen Welt) und muss global eindeutig sein. Die einzige Ausnahme gibt es bei Mehrfachrepräsentationen. Dann dürfen die zugehörigen Repräsentationen bei verschiedenen Datenanbietern bzw. in verschiedenen Datensätzen die gleiche Objekt-ID verwenden, um damit explizit die Zusammengehörigkeit der Repräsentationen zu kennzeichnen. Innerhalb eines Datensatzes muss eine Objekt-ID auf jeden Fall eindeutig sein. Im Nexus-System sind die Datensatz-ID und die Objekt-ID jeweils 128 Bit lang und sind nach dem Prinzip der UUIDs (Universally Unique IDentifier, [LMS05]) aufgebaut, so dass die irrtümliche Zuordnung

gleicher IDs extrem unwahrscheinlich ist. Das Prinzip der UUIDs stellt sicher, dass Datenanbieter unabhängig voneinander und ohne sich absprechen zu müssen eindeutige Objekt-IDs erzeugen können. Die IDs im Beispiel sind diesbezüglich vereinfacht.

Durch die Kombination des NOL aus Servername, Datensatz-ID und Objekt-ID kann immer eindeutig der Ursprung der Daten bestimmt werden und gegebenenfalls der Datenanbieter direkt angesprochen werden, um weitere Daten bzw. Attribute des Objekts nachzuladen. Die Konstruktion macht zudem einen Verzeichnisdienst unnötig, der zu einem gegebenen NOL den Datenanbieter zurückliefert, bei dem das zugehörige Objekt gespeichert ist. Eine Anwendung kann die Daten des Objekts zu einem gegebenen NOL erhalten, ohne dass sie wissen muss, wo diese Daten gespeichert sind, indem sie eine entsprechende Anfrage an die Föderationskomponente stellt. Die Föderationskomponente kann durch den Aufbau des NOL direkt den gewünschten Datenanbieter ansprechen. Eine Migration eines Objekts zu einem anderen Datenanbieter ist im Nexus-System nicht vorgesehen. Hierfür müssten alle aktuellen Nutzer des Objekts notifiziert werden, oder ein automatischer Umleitungsmechanismus etabliert werden. Stattdessen erhält ein migriertes Objekt vom neuen Datenanbieter einen neuen NOL, bei dem die Objekt-ID übernommen wird. Nachdem das Objekt über eine räumliche Anfrage gefunden wurde kann mit der neuen NOL wieder direkt darauf zugegriffen werden.

Um neu aufgekommenen Anforderungen der Projektpartner gerecht zu werden, wurde in [HKNS05] das Datenmodell wie folgt erweitert: Jedes Attribut besteht nicht mehr nur aus einem einzelnen Wert, sondern kann, wie im Beispiel in Abbildung 3 das Ausstellungsattribut des Museums, aus mehreren Teilen bestehen. Der Verbund aus den jeweils zusammengehörigen Teilen eines Attributs wird Attributinstanz genannt. Diese Erweiterung wurde insbesondere deshalb eingeführt, um zu jedem Attributwert noch zusätzliche Metadaten wie Erfassungszeit (*observationTime*) oder Gültigkeitszeit (*validTime*) speichern zu können. Mit den gleichen Mechanismen können nun aber auch komplex strukturierte Attributwerte realisiert werden. Das Ausstellungsattribut des Museums besteht aus den Attributteilen *value.Name* und *value.Thema* (der Ausstellung) sowie dem Metaattributteil *meta.validTime* (Gültigkeitszeit). In letzterem Attributteil wird die Dauer der Ausstellung gespeichert. Attribute, die auch weiterhin nur aus einem Wert bestehen, speichern diesem Wert im Attributteil namens *value*. Die Daten des Museumsobjekts sind nun so organisiert, wie es in Abbildung 4 angedeutet wird. Der Name eines Attributteils ist eindeutig innerhalb einer Attributinstanz, gleiche Attributteile können nur in verschiedenen Attributinstanzen auftreten, also bei Mehrfachattributen.

Das Schema des Umgebungsmodells besteht derzeit aus ca. 250 Objekttypen. Das Schema ist untergliedert in einen Standardteil, der das gemeinsame Fundament für alle Anwendungen und Datenanbieter bildet, und in beliebig viele Erweiterungen. Der Standardteil enthält alle Objekttypen, die nach der Analyse aus [Mess01] typischerweise von ortsbezogenen Anwendungen verwendet werden. Falls diese Objekttypen nicht ausreichen, kann ein Datenanbieter eine eigene Schemaerweiterung erstellen. Die dort definierten Objekttypen müssen direkt oder indirekt von einem Objekttyp des Standardteils erben. Auf diese Weise kann der standardkonforme Teil der erweiterten Daten des Anbieters auch von Anwendungen genutzt werden, die dessen Erweiterung nicht kennen. Es ist anzustreben, dass die Erweite-

Objekt	Attribut	Attributteil	Wert	
o1	NOL	value	nexus:http://nexus.uni-stuttgart.de||0x57188c15/0x79990559	
	Typ	value	Museum	Attributinstanz
	Name	value	Linden-Museum	Attributinstanz
	Position	value	POINT(48.7826 9.1700)	
	Grundriss	value	POLYGON ((48.782393 9.1701981, 48.782534 9.169604, 48	
	Ausstellung	value.Name	Die Kultur der Inka	
		value.Thema	Völkerkunde	Attributinstanz
		meta.validTime	begin: 2006-05-01 end: 2006-07-31	
	Ausstellung	value.Name	Der Alltag der Neandertaler	
		value.Thema	Naturkunde	Attributinstanz
		meta.validTime	begin: 2006-08-01 end: 2006-10-31	

Abbildung 4: Datenstruktur des Beispielobjekts

rungen koordiniert entwickelt werden, so dass mehrere Datenanbieter die selbe Erweiterung nutzen und dadurch deren Daten interoperabel sind. Das Schema des Umgebungsmodells wird ausführlich in [Nick05] diskutiert.

3

Föderationssansatz

In diesem Kapitel wird der grundlegende Architekturansatz für die Verarbeitung von ortsbezogenen Anfragen in einer Föderation aus unabhängigen, lose gekoppelten Datenanbietern vorgestellt. Eine detaillierte Diskussion der Konzepte zur föderierten Verarbeitung von ortsbezogenen Anfragen folgt in den nächsten beiden Kapiteln. Zunächst wird der für die Anfrageverarbeitung relevante Teil der Architektur des Nexus-Systems in Kapitel 3.1 erläutert. Es folgen eine Beschreibung des Datenaustauschformats in Kapitel 3.2 und der Anfragesprache in Kapitel 3.3. Beide werden sowohl von einer Anwendung bei der Kommunikation mit einer Föderationskomponente als auch von einer Föderationskomponente bei der Kommunikation mit einem Datenanbieter verwendet. Die Funktionsweise und mögliche Implementierungen des Verzeichnisdienstes werden in Kapitel 3.4 präsentiert. Schließlich wird in Kapitel 3.5 die Nutzbarkeit alternativer Architekturansätze für die Integrationsmiddleware abgewogen.

3.1 Basisarchitektur

Die Architektur des Nexus-Systems [NGS+01, SHG+04, MNG+05] besteht aus drei Ebenen, aus der Ebene der Datenanbieter, aus der Ebene der Integrationsmiddleware sowie aus der Ebene der Anwendungen, wovon die meisten auf mobilen Endgeräten ablaufen. Die Architektur ist in Abbildung 5 dargestellt.

Ein Datenanbieter, welcher im Nexus-Jargon auch Kontextserver genannt wird, speichert ein lokales Umgebungsmodell, welches einen Ausschnitt der realen Welt aus dem Blickwinkel des Anbieters nachbildet. Ein Kontextserver ist vergleichbar mit einem Webserver im WWW. Damit ein Datenanbieter in das Nexus-System integriert werden kann muss er zwei Voraussetzungen erfüllen. Erstens muss er eine Anfrageschnittstelle bereitstellen, welche die in Kapitel 3.3 beschriebene Anfragesprache als Eingabe akzeptiert und welche ihre Ergebnisse in dem Datenaustauschformat zurückliefert, das in Kapitel 3.2 beschrieben wird. Diese erste Voraussetzung impliziert zudem, dass die Daten des Anbieters gemäß dem Schema des Umgebungsmodells strukturiert sind. Zweitens muss sich der Datenanbieter mit seinem Dienstgebiet und den Typen der gespeicherten Objekte beim Verzeichnisdienst anmelden. Nur dann wird er auch bei der Verarbeitung von Anfragen

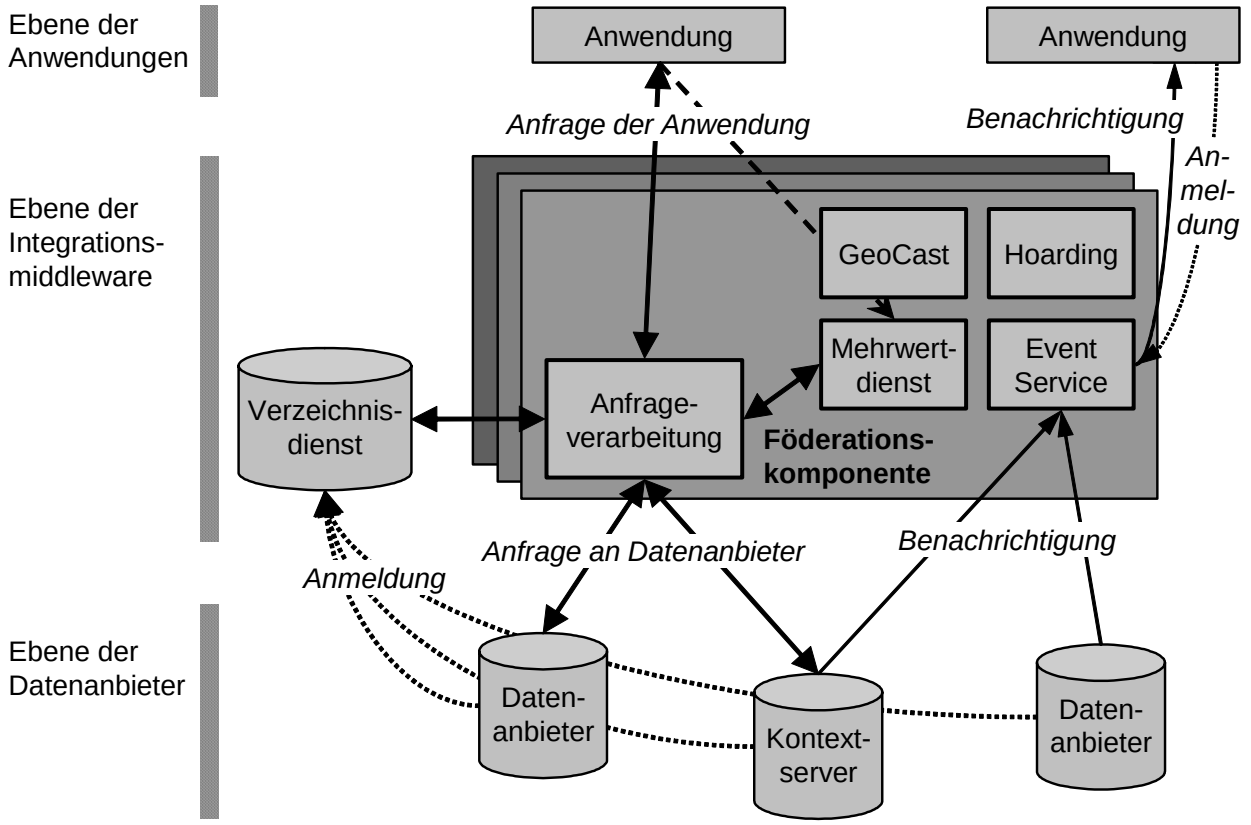


Abbildung 5: Architektur der Nexus-Plattform

berücksichtigt. Falls der Datenanbieter auch dem Event Service zur Verfügung stehen soll, und diesen beim Eintreten gegebener Zustände benachrichtigen soll, so muss er weitere Schnittstellen bereitstellen, die jedoch nicht Gegenstand dieser Arbeit sind.

Je nach den Eigenschaften der lokal gespeicherten Daten kann ein Datenanbieter auf verschiedene Arten implementiert sein [GBH+05]. Große Geodatenmodelle wie beispielsweise Stadtmodelle oder Straßennetze werden am besten mit Hilfe eines Datenbanksystems, das mit räumlichen Daten umgehen kann, verwaltet. Die sich ständig ändernden Positionen der mobilen Benutzer, die damit eine hohe Aktualisierungsrate aufweisen, werden am besten mit einem verteilten Hauptspeichersystem verwaltet [LeRo02]. Die Daten eines Eigenheims, das spezielle kontextbezogene Computerunterstützung für allein lebende Senioren bietet, können mit einem leichtgewichtigen Server bereitgestellt werden, da sie nur ein geringes Volumen und auch nur einen geringen Relevanzbereich aufweisen [LBBN04]. Selbst einzelne Sensoren auf einer Sensorplattform wie dem ContextCube können als Datenanbieter im Nexus-System teilnehmen [BBHS03]. Weitere Implementierungen und die Charakteristika der jeweiligen Implementierung können in [GBH+05] nachgelesen werden.

Die Integrationsmiddleware besteht aus der Föderationskomponente und dem Verzeichnisdienst, siehe Abbildung 5. Die Föderationskomponente vermittelt zwischen den Anwendungen und den Datenanbietern. Da sie selbst keine Daten speichert, abgesehen von Caching, kann sie leicht vervielfältigt werden, um somit mehr Benutzer bedienen zu können. Die Föderationskomponente besitzt die gleiche Anfrageschnittstelle wie ein Datenanbieter. Es ist somit auch denkbar, dass eine

Föderationskomponente gegenüber einer anderen (übergeordneten) Föderationskomponente als ein eigenständiger Datenanbieter auftritt, der die untergeordneten Datenanbieter zusammenfasst [DSB+04]. Zusätzlich zur Anfrageverarbeitung beherbergt die Föderationskomponente weitere Funktionalitäten [SHG+04], die in dieser Arbeit jedoch nur von geringer Bedeutung sind. Zu diesen Funktionalitäten gehören der Event Service, GeoCast, Hoarding und Mehrwertdienste. Der Event Service [BaRo04] ermöglicht ein anderes Datenflussparadigma. Dort erhält eine Anwendung die Daten nicht auf eine konkrete Anfrage hin, sondern immer dann, wenn ein von der Anwendung definierter Zustand eintritt. Beispielsweise kann eine Anwendung den Benutzer benachrichtigen, sobald er sich in der Nähe eines Bekannten oder in der Nähe eines Ladens aufhält, der einen gesuchten Artikel verkauft. GeoCast [DüRo03] erlaubt das Verschicken von Nachrichten an alle mobilen Nutzer, die sich in einem gegebenen Gebiet aufhalten. Hoarding [BMR04] ermöglicht den entkoppelten Betrieb eines mobilen Endgeräts, indem vorab die vermutlich bald benötigten Daten auf das Endgerät geladen werden, solange es noch mit der Infrastruktur (also der Integrationsmiddleware) verbunden ist. Mehrwertdienste haben eine proprietäre Schnittstelle und erlauben das Verlagern von häufig benötigter Funktionalität in die Infrastruktur, um die mobilen Endgeräte zu entlasten. Zusätzlich können Mehrwertdienste viel effizienter auf die Daten des Umgebungsmodells zugreifen, da sie direkt in der Föderationskomponente ausgeführt werden. Typische Mehrwertdienste sind der Navigationsdienst, der einen Weg zwischen einem gegebenen Start- und Endpunkt berechnet, sowie der Kartendienst, der Straßen und Gebäude sowie weitere interessante Orte in eine Karte einzeichnet und diese als Vektorgrafik oder Pixelbild an die Anwendung liefert.

Ein weiterer Teil der Integrationsmiddleware ist der Verzeichnisdienst, welcher den Überblick über alle aktuell verfügbaren Datenanbieter behält. Hierfür müssen sich die Datenanbieter beim Verzeichnisdienst an- und abmelden. Die Föderation benutzt den Verzeichnisdienst, um zu einer gegebenen Anfrage die relevanten Datenanbieter herauszufinden. Der Verzeichnisdienst wird in Kapitel 3.4 näher erläutert.

Die orts- oder kontextbezogenen Anwendungen laufen typischerweise auf mobilen Endgeräten. Sie können auf die Nexus-Plattform auf drei verschiedene Arten zugreifen. Erstens können sie räumliche Anfragen an die Föderationskomponente schicken, um integrierte Umgebungsmodelldaten über ihre Umgebung zu erhalten. Zweitens können sie Ereignisse bzw. deren beschreibende Prädikate beim Event Service registrieren, um dann benachrichtigt zu werden, wenn die Ereignisse eintreten, beispielsweise wenn ein Benutzer ein Gebäude betritt, oder wenn die Raumtemperatur einen Schwellwert übersteigt. Drittens können sie Mehrwertdienste benutzen, um die Umgebungsmodelldaten schon in verarbeiteter Form, beispielsweise als Karte, zu erhalten.

Die einzelnen Komponenten des Nexus-Systems kommunizieren miteinander über das SOAP Protokoll (Simple Object Access Protocol, [Box00]). Dieses hat den Vorteil, dass die einzelnen Komponenten in verschiedenen Programmiersprachen implementiert werden können, und dass vor allem die transportierten Daten sehr flexibel strukturiert werden können. Dies ist insbesondere wichtig, um Mehrfachtypen und Mehrfachattribute handhaben zu können, siehe Kapitel 2.3 und 3.2.

3.1.1 Integrationsprinzip: Föderation versus Integration

Um die Daten mehrerer Anbieter zu kombinieren sind zwei verschiedene Ansätze denkbar: der Föderationsansatz und der Integrationsansatz. Es werden keine Ansätze betrachtet, die eine Anpassung der Datenanbieter erfordern, wie beispielsweise eine Änderung des Schemas oder der eingesetzten Datenhaltungssoftware.

Beim Föderationsansatz verbleiben alle Daten an ihrem ursprünglichen Speicherort. Die Föderationskomponente leitet eingehende Anfragen an die relevanten Datenanbieter weiter, sammelt deren Antworten ein und integriert diese zu einer Gesamtantwort, die an die Anwendung zurückgeschickt wird. Schema- und Schnittstellenheterogenitäten werden von sogenannten Wrappern angepasst. Dateninkonsistenzen werden beim Zusammenführen der Anworten der Datenanbieter aufgelöst, oder auch an die Anwendung weitergeleitet, falls dies auf die Schnelle nicht möglich ist.

Beim Integrationsansatz werden alle Daten in ein zentrales Data Warehouse kopiert, das anschließend als einzige Datenquelle weiterverwendet wird. Schema- und Schnittstellenheterogenitäten sowie Dateninkonsistenzen können beim Beladen des Warehouses gelöst werden.

Der Vorteil des Integrationsansatzes ist, dass keine komplexen Datentransformationen zur Laufzeit durchgeführt werden müssen, was zu kurzen und vorhersagbaren Antwortzeiten führt. Jedoch kann so ein Warehouse sehr leicht sehr groß werden, da insbesondere räumliche Daten tendenziell voluminös sind. Die Datenanbieter geben ihre Daten aus der Hand und haben keine Kontrolle mehr über den Zugriff darauf. Zudem erfordert das Hinzufügen neuer Datenanbieter oder das Entfernen von Datenanbietern, die ihren Dienst einstellen, unter Umständen große Änderungen im Warehouse. Alle Aktualisierungen, die beispielsweise von Sensoren kommen, müssen in einem einzigen Warehouse nachgeführt werden, das schnell zum Flaschenhals wird.

Die Vorteile des Föderationsansatzes sind die Nachteile des Integrationsansatzes, und umgekehrt. Die Daten verbleiben bei ihren Anbietern, neue Anbieter können leicht zum System hinzugefügt werden und jeder Datenanbieter kann seinen lokalen Datenbestand selbst aktualisieren. Demgegenüber müssen die Daten verschiedener Anbieter während der Anfrageverarbeitung integriert werden, was möglicherweise komplexe, und damit rechenintensive, semantische Datenintegrationsverfahren erfordert. Die Antwortzeit von Anfragen verlängert sich schon allein dadurch, dass die Anfragen zunächst an die relevanten Datenanbieter weitergeleitet werden müssen. Wenn ein Datenanbieter kurzzeitig ausfällt, dann steht auch keine Sicherheitskopie seiner Daten zur Verfügung, wie dies in einem Warehouse der Fall wäre.

Im Nexus-System wird der Föderationsansatz verfolgt, da dieser besser mit sich häufig ändernden Konfigurationen von Datenanbietern zurecht kommt [MNG+05]. Zudem gibt er den Datenanbietern mehr Flexibilität und mehr Kontrolle über ihre Daten. Schließlich können Aktualisierungen dezentral verarbeitet werden.

3.1.2 Föderationsmethode: Lokal-als-Sicht versus Global-als-Sicht

Der Zusammenhang zwischen dem lokalen Schema eines Datenanbieters und dem globalen Schema der Integrationsmiddleware kann auf verschiedene Arten festgelegt werden. Die wichtigsten Arten sind Global-als-Sicht (engl. global-as-view, GAV) [GPQ+97] und Lokal-als-Sicht (engl. local-as-view, LAV) [LRO96].

Beim GAV-Ansatz wird ein Element des globalen Schemas als eine Sicht über ein oder mehrere Elemente aus den lokalen Schemata der Datenanbieter definiert. Einerseits können somit Anfragen sehr effizient verarbeitet werden, da direkt aus der Sichtendefinition abgelesen werden kann, an welche Datenanbieter eine Anfrage weitergeleitet werden muss. Andererseits muss die Definition des globalen Schemas jedes Mal angepasst werden, wenn neue Datenanbieter hinzukommen oder wenn alte verschwinden.

Der LAV-Ansatz verfolgt ein umgekehrtes Konzept. Ein Element des lokalen Schemas eines Datenanbieters wird definiert als eine Sicht über ein oder mehrere Elemente des globalen Schemas. Somit können neue Datenanbieter hinzugefügt oder bekannte Datenanbieter entfernt werden ohne dass das globale Schema angepasst werden muss. Jedoch muss bei der Anfrageverarbeitung jetzt das Problem der „Beantwortung von Anfragen mit Sichten“ gelöst werden, welches bekannterweise NP-hart ist [Hale01].

Im Nexus-System wird ein vereinfachter LAV-Ansatz verwendet, weil dieser leichter mit dynamischen Konfigurationen von Datenanbietern zurecht kommt [MNG+05]. Der vereinfachte Ansatz verlangt jedoch von den Datenanbietern, dass sie ihre Daten nur im Format des globalen Schemas anbieten dürfen, und dass sie sich selbst um die notwendigen Daten- und Schematransformationen kümmern. Dies ermöglicht nun, dass ein Element (ein Objekttyp) des lokalen Schemas genau einem Element des globalen Schemas zugeordnet wird, nämlich demselben Objekttyp im globalen Schema. Umgekehrt kann deshalb zur Beantwortung von Anfragen ein Element des globalen Schemas direkt auf genau ein Element der lokalen Schemata der Datenanbueter abgebildet werden, wodurch die NP-Härte aufgebrochen wird. Zusätzlich kann ein Datenanbieter den Wertebereich von Attributen der bei ihm gespeicherten Objekte vorgeben, wobei dies derzeit nur mit dem Typ- und dem Positionsattribut möglich ist. So kann ein Anbieter vorgeben, dass er nur Hotels im Stadtgebiet Stuttgarts speichert. Diese Einschränkungen der Wertebereiche werden im Verzeichnisdienst verwaltet, siehe Kapitel 3.4. Diese Vereinfachung ermöglicht es, dass Anfragen ohne einen NP-vollständigen Transformationsschritt verarbeitet werden können.

3.2 Datenaustauschformat

Für den Transport der Umgebungsmodelldaten zwischen den Komponenten des Nexus-Systems wurde das Datenaustauschformat AWML (Augmented World Modeling Language) entwickelt, das speziell auf die Charakteristika des in Kapitel 2.3 beschriebenen Datenmodells abgestimmt ist. Es kann mit Mehrfachtypen und mit Mehrfachattributen umgehen, und es erlaubt zudem den Transport von Objekten unterschiedlichen Typs in einer einzigen Ergebnismenge. AWML basiert auf XML, jedoch mussten das logische und das physische Schema getrennt

```
<?xml version='1.0' encoding='UTF-8'?>
<awml:awml xmlns:gml="http://www.opengis.net/gml" xmlns:awml="http://www.nexus.uni-stuttgart.de/2.0/AWML"
xmlns:nsat="http://www.nexus.uni-stuttgart.de/1.0/NSAT" xmlns:nsas="http://www.nexus.uni-stuttgart.de/1.0/NSAS"
xmlns:nscs="http://www.nexus.uni-stuttgart.de/1.0/NSCS">
 <awml:nexusobject>
  <nsas:NOL> <nsas:value>
     nexus:http://nexus.uni-stuttgart.de/l0x57188c15/0x79990559
  </nsas:value> </nsas:NOL>
  <nsas:Typ> <nsas:value>nscs:Museum</nsas:value> </nsas:Typ>
  <nsas:Grundriss> <nsas:value><nsat:WKT srscode="4326">
MULTIPOLYGON (((48.782393 9.1701981, 48.782534 9.169604, 48.7828949 9.1697999, 48.78307
9.170104, 48.782823 9.1704271, 48.782701 9.170364, 48.782641 9.1704059, 48.78261 9.1703141,
48.782393 9.1701981),
(48.7827701 9.170014, 48.782704 9.1702889, 48.782812 9.1703469, 48.7829151 9.170211,
48.782815 9.170037, 48.7827701 9.170014)))
  </nsat:WKT> </nsas:value> </nsas:extent>
  <nsas:kind> <nsas:value>real</nsas:value> </nsas:kind>
  <nsas:Name> <nsas:value>Linden-Museum</nsas:value> </nsas:name>
  <nsas:Position> <nsas:value><nsat:WKT srscode="4326">
     POINT(48.7826  9.1700)
  </nsat:WKT> </nsas:value> </nsas:Position>
  <nsas:Ausstellung>
   <nsas:value>
    <nsas:Name>Die Kultur der Inka</nsas:Name>
    <nsas:Thema>Völkerkunde</nsas:Thema>
   </nsas:value>
   <nsas:meta> <nsas:validTime> <gml:TimePeriod>
      <gml:begin> <gml:TimeInstant> <gml:timePosition>
         2006-05-01T
      </gml:timePosition> </gml:TimeInstant> </gml:begin>
      <gml:end> <gml:TimeInstant> <gml:timePosition>
         2006-07-31T
      </gml:timePosition> </gml:TimeInstant> </gml:end>
   </gml:TimePeriod> </nsas:validTime> </nsas:meta>
  </nsas:Ausstellung>
  <nsas:Ausstellung>
   <nsas:value>
    <nsas:Name>Der Alltag der Neandertaler</nsas:Name>
    <nsas:Thema>Naturkunde</nsas:Thema>
   </nsas:value>
   <nsas:meta> <nsas:validTime> <gml:TimePeriod>
      <gml:begin> <gml:TimeInstant> <gml:timePosition>
         2006-08-01T
      </gml:timePosition> </gml:TimeInstant> </gml:begin>
      <gml:end> <gml:TimeInstant> <gml:timePosition>
         2006-10-31T
      </gml:timePosition> </gml:TimeInstant> </gml:end>
   </gml:TimePeriod> </nsas:validTime> </nsas:meta>
  </nsas:Ausstellung>
 </awml:nexusobject>
</awml:awml>
```

Abbildung 6: Museumsobjekt in AWML kodiert

werden, um die angegebenen Anforderungen erfüllen zu können. Das physische Schema beschreibt den Aufbau eines AWML-Dokuments und wird in [BDG+04] als XML-Schema spezifiziert. Es definiert generische Objekte, die alle möglichen Attribute in beliebiger Anzahl und Kombination enthalten dürfen. Auch der Typ eines Objekts wird wie ein Attribut repräsentiert. Dies ist notwendig, um Mehrfachtypen unterstützen zu können. Dies hat zur Folge, dass allein mit dem physischen Schema nicht überprüft werden kann, ob die Attribute eines Objekts zu dessen Objekttyp passen, und ob alle vorgeschriebenen Attribute enthalten sind. Hierfür ist das logische Schema notwendig, welches unter anderem in [Nick05] beschrieben ist, und welches die Vererbungsbeziehungen der Objekttypen und die zugehörigen Attribute festlegt. Die Überprüfung des logischen Schemas übernimmt das Anwendungsprogramm bzw. der Parser, der das Datenaustauschformat einliest. Das logische Schema liegt ebenfalls in XML-Schema-Notation vor. Dort sind jedoch keine Mehrfachtypen möglich. Diese Einschränkung stellt kein Problem dar, da das logische Schema nicht von einem validierenden XML Parser benutzt wird sondern vom Anwendungsprogramm. Sie unterstreicht aber die Notwendigkeit der Trennung von physischem und logischem Schema.

Durch die Verwendung von XML stellen Mehrfachattribute kein Problem dar. Da Attribute durch XML Elemente repräsentiert werden, können Mehrfachattribute durch mehrere Instanzen des selben XML Elements dargestellt werden. Komplexe Attribute, die aus mehreren Attributteilen und Metaattributteilen[1] bestehen, können ebenso repräsentiert werden. Dadurch, dass AWML nur generische Objekte kennt, können auf triviale Weise die Objekte einer AWML Ergebnismenge alle möglichen Typen haben. Abbildung 6 zeigt das Museumsobjekt aus Abbildung 4 in Kapitel 2.3 im AWML Format. Die Repräsentation der Daten als XML Dokument hat den weiteren Vorteil, dass dies gut mit dem SOAP Protokoll harmoniert.

3.3 Anfragesprache

Die Anfragesprache AWQL (Augmented World Query Language) ist ebenfalls speziell auf die Anforderungen der kontextbezogenen Anwendungen zugeschnitten. Sie erlaubt im Wesentlichen die Selektion und die Projektion von Objekten. Eine Untersuchung zu Anfang des Nexus-Projekts [Nick00] hat ergeben, dass orts- und kontextbezogene Anwendungen sehr gut mit diesem Funktionsumfang einer Anfragesprache zurecht kommen können. Solche Anwendungen benötigen keine volle SQL Mächtigkeit, wodurch die Implementierung aller Anfragen verarbeitenden Komponenten wesentlich vereinfacht werden kann.

Dadurch, dass die Objekttypen im Umgebungsmodell als Attribute dargestellt werden, befinden sich alle Objekte in einer einzigen riesigen Menge, die an anderer Stelle auch als universelle Relation bezeichnet wird. Eine Tabelle wie in der FROM Klausel bei SQL muss nicht explizit ausgewählt werden. Stattdessen wird der Objekttyp in der Selektion über ein Vergleichsprädikat mit dem Typattribut eingeschränkt. Dieser Vergleich wählt implizit auch alle Subtypen des angegebenen Typs

[1] Die Gruppierung der normalen Attributteile in einem `<value>` Tag und der Metaattributteile in einem `<meta>` Tag liegt in der historischen Evolution des Datenaustauschformats begründet, unterstreicht jedoch auch die semantischen Unterschiede zwischen den Attributteilen.

```
<?xml version="1.0" encoding="UTF-8"?>
<awql:awql xmlns:awql="http://www.nexus.uni-stuttgart.de/2.0/AWQL"
xmlns:nsat="http://www.nexus.uni-stuttgart.de/1.0/NSAT" xmlns:nsas="http://www.nexus.uni-stuttgart.de/1.0/NSAS"
xmlns:nscs="http://www.nexus.uni-stuttgart.de/1.0/NSCS" xmlns:gml="http://www.opengis.net/gml">
 <awql:restriction>
  <awql:and>
   <awql:equal>
    <awql:target>nsas:Typ.nsas:value</awql:target>
    <awql:referenceValue>nscs:Museum</awql:referenceValue>
   </awql:equal>
   <awql:intersects>
    <awql:target>nsas:Position.nsas:value</awql:target>
    <awql:referenceValue> <nsat:WKT srscode="4326">
       POLYGON ((48.7948 9.1506, 48.7948 9.2097, 48.7653 9.2097, 48.7653 9.1505, 48.7948 9.1506))
    </nsat:WKT> </awql:referenceValue>
   </awql:intersects>
   <awql:or>
    <awql:equal>
     <awql:target>nsas:Ausstellung.nsas:value</awql:target>
     <awql:referenceValue>Naturkunde</awql:referenceValue>
    </awql:equal>
    <awql:equal>
     <awql:target>nsas:Ausstellung.nsas:value</awql:target>
     <awql:referenceValue>Völkerkunde</awql:referenceValue>
    </awql:equal>
   </awql:or>
  </awql:and>
 </awql:restriction>
 <awql:filter>
  <awql:include>
   <awql:target>nsas:Name</awql:target>
   <awql:include><awql:target>nsas:value</awql:target></awql:include>
  </awql:include>
  <awql:include>
   <awql:target>nsas:Position</awql:target>
   <awql:include><awql:target>nsas:value</awql:target></awql:include>
  </awql:include>
  <awql:include>
   <awql:target>nsas:Ausstellung</awql:target>
   <awql:restriction>
    <awql:temporalIntersects>
     <awql:target>
      nsas:Ausstellung.nsas:meta.nsas:validTime
     </awql:target>
     <awql:referenceValue> <gml:TimeInstant> <gml:timePosition>
        2006-09-06T16:45:37
     </gml:timePosition> </gml:TimeInstant> </awql:referenceValue>
    </awql:temporalIntersects>
   </awql:restriction>
  </awql:include>
 </awql:filter>
</awql:awql>
```

Abbildung 7: AWQL Beispielanfrage

aus. Weiterhin stehen räumliche Vergleichsprädikate (*intersects*, *within*), ein exakter Vergleich (*equal*), ein spezieller Vergleich für Zeichenketten (*like*) sowie größer und kleiner Vergleiche (*greater*, *less*) zur Verfügung, die beliebig boolesch (*and*, *or*, *not*) verknüpft werden können. Die Selektion wird durch den *restriction* Teil in der AWQL Anfrage repräsentiert. Eine typische Selektionsanfrage möchte alle Objekte mit bestimmten Eigenschaften in einem gegebenen Kartenausschnitt darstellen („Zeichne ein Piktogramm an den Stellen in die Karte ein, an denen sich ein Naturkundemuseum befindet."). Die allermeisten Anfragen in der Domäne der kontextbezogenen Anwendungen enthalten mindestens ein räumliches Prädikat sowie ein Prädikat auf dem Typattribut.

Wenn anstatt einer normalen Anfrage eine Nachbarschaftsanfrage gestellt werden soll, so muss die AWQL Anfrage um einen *nearestNeighbor* Teil ergänzt werden. Dieser definiert den Referenzpunkt der Nachbarschaftsanfrage, die Anzahl der gewünschten Ergebnisobjekte, sowie das Attribut auf dem der Vergleich durchgeführt werden soll. Typischerweise ist dies das *pos* Attribut. Die Prädikate aus dem *restriction* Teil sind weiterhin wirksam, und erlauben somit, die Menge der qualifizierenden Objekte einzuschränken. Eine typische Nachbarschaftsanfrage möchte die nächstgelegenen Objekte mit bestimmten Eigenschaften ermitteln („Welches sind die drei nächstgelegenen Naturkundemuseen?").

Bei der Projektion können entweder bestimmte Attribute ausgewählt (*include*) oder ausgeschlossen (*exclude*) werden. Dies bestimmt, welche Attribute der Objekte in der zurückgelieferten Ergebnismenge enthalten sind. Zusätzlich können die zurückgelieferten Attributinstanzen eingeschränkt werden, so dass nicht alle Instanzen geliefert werden sondern nur ausgewählte. Dies ist beispielsweise dann sinnvoll, wenn ein kleiner Ausschnitt aus einer langen Positionshistorie angefordert werden soll (Wo war ich gestern zwischen 14 und 16 Uhr?). In einer AWQL Anfrage können deshalb innerhalb der Projektion zusätzliche Prädikate angegeben werden, welche die zurückgelieferten Attributinstanzen einschränken. Die Projektion wird durch den *filter* Teil in der AWQL Anfrage repräsentiert.

Eine AWQL Anfrage kann weitere Teile enthalten, die hier nur am Rande erwähnt werden sollen. Es können die Schemaerweiterungen genannt werden, welche die Anwendung kennt. Objekte, die keinem der genannten Schemata angehören, werden automatisch in ein Objekt des nächstgelegenen Typs aus dem Standardschema umgewandelt. Eine Anwendung kann gezielt die zu fragenden Datenanbieter vorgeben und damit die Anfrage beim Verzeichnisdienst ersetzen. Es kann das Format zur Repräsentation der räumlichen Attribute in der Ergebnismenge sowie das dabei verwendete Koordinatensystem vorgeben werden. Des Weiteren bietet die Anfragesprache auch Möglichkeiten zum Aktualisieren und Löschen von Objekten.

Die Beispielanfrage in Abbildung 7 sucht nach allen Museen in einem gegebenen Kartenausschnitt, die gerade eine Ausstellung zum Thema Naturkunde oder zum Thema Völkerkunde bieten. Vom Ausstellungsattribut sollen nur die Instanzen zurückgeliefert werden, die zum Anfragezeitpunkt relevant sind.

3.4 Verzeichnisdienst

Ein Verzeichnisdienst wird in der Regel dazu eingesetzt, um den Zugriff auf Daten, Dienste oder Ressourcen zu virtualisieren. Mit Hilfe des Verzeichnisdienstes muss eine Anwendung nur beschreiben, was sie will, aber nicht, wo genau es zu finden ist, oder wie darauf zugegriffen werden muss. Der Verzeichnisdienst übersetzt die vage Beschreibung der Anwendung in präzise Verweise auf die gewünschten bzw. relevanten Ressourcen. Web Services [CDK+02] und Grid Computing [FoKe99] stellen zwei bekannte Beispiele dar, die eine solche Virtualisierung anstreben.

Im Nexus-System soll der Verzeichnisdienst den Überblick über die aktuell verfügbaren Datenanbieter behalten. Für die Föderation soll er zu einer gegebenen Anfrage die relevanten Datenanbieter ermitteln. Hierfür speichert der Verzeichnisdienst von jedem Datenanbieter eine zusammenfassende Inhaltsbeschreibung der verwalteten Objekte. Auf Grund der Charakteristika der typischen Anfragen von ortsbezogenen Anwendungen, welche die gesuchten Objekte meist nach Typ und Ort einschränken, hat es sich als vorteilhaft erwiesen, die Inhaltsbeschreibung aus diesen beiden Attributen zusammenzusetzen. Für jeden Datenanbieter wird eine Liste der Objekttypen der verwalteten Objekte sowie ein Dienstgebiet, das die Geometrien aller verwalteten Objekte einschließt, gespeichert. Folglich kann der Verzeichnisdienst auch als räumliches Verzeichnis bezeichnet werden. Zusätzlich speichert der Verzeichnisdienst weitere Informationen über die Fähigkeiten eines Datenanbieters, beispielsweise, ob dieser Nachbarschaftsanfragen verarbeiten kann, ob dieser die Objekte in einer Ergebnismenge nach bestimmten Kriterien sortieren kann oder welche Version der Schnittstellensprache dieser unterstützt.

Der Verzeichnisdienst bietet zwei Anfrageschnittstellen an. Erstens gibt er zu einem gegebenen Gebiet und einem gegenen Typ eine Liste aller Datenanbieter zurück, deren Dienstgebiet das gegebene Gebiet schneidet, und in deren Typliste sich der gegebene Typ oder ein Subtyp davon befindet. Mit solchen Anfragen ermittelt die Föderation die relevanten Anbieter. Zweitens liefert der Verzeichnisdienst die weiteren Informationen zu einem Datenanbieter, dessen Name bzw. Datensatzidentifikator gegeben wird. Dies kann der Föderation helfen, den richtigen Anfrageplan zu wählen.

Derzeit gibt es schon eine Fülle von Verzeichnisdiensten für verschiedene Einsatzszenarien. Es gibt LDAP-basierte Produkte [Nove01], CORBA Trader [OMG00], Implementierungen des UDDI [Oasis04] Standards, die Web Services verwalten, sowie Grid Information Services [CFFK01, RWM02, SWMY00]. Auch gewöhnliche Websuchmaschinen [BrPa98] können hierzu gezählt werden. Alle bieten sie hochentwickelte Suchmöglichkeiten, die auf Schlüsselwörtern, Hierarchien oder Attributvergleichen basieren, aber keiner dieser Verzeichnisdienste berücksichtigt eine räumliche Dimension, um Ergebnisse zu selektieren oder anzuordnen. Da das Nexus-System durch seine Offenheit auch eine sehr große Anzahl an Datenanbietern erwartet, kann ein reiner Schlüsselwort- oder Attribut-Wert-Vergleich-basierter Suchansatz die Anzahl der als relevant eingeschätzen Datenanbieter nicht ausreichend einschränken. Eine Abbildung der räumlichen Dimension in eine geographische Taxonomie (beispielsweise mit Hierarchiestufen wie Land, Bundesland, Stadt, Stadtteil) ist nicht feingranular genug. Der Ortsbezug und die räumliche Anordnung der Datenanbieter ist ein probates Mittel, um das Suchergebnis zu verfeinern. Es sollen ja nicht irgendwelche Hotelanbieter gefunden werden, sondern

nur welche, die auch für den gegebenen Kartenausschnitt relevant sind. In der Smart Factory beispielsweise kann somit nicht nur ein passender Lieferant für benötigte Teile gefunden werden, sondern zusätzlich auch noch ein naheliegender, so dass Transportwege und Reaktionszeiten kurz bleiben.

Am anderen Ende des Spektrums gibt es einige geographischen Verzeichnisdienste wie das GeoWeb [LRIE01] oder Implementierungen [CoTe03] der OGC Catalog Services Specification [OGC02], die einer Anwendung erlauben, passende Ressourcen nach deren Lokation zu finden. Natürlich können auch weitere Vergleiche zur Einschränkung der passenden Ressourcen eingesetzt werden. Was jedoch fehlt, ist die Klassifikation der Ressourcen entlang einer Konzepthierarchie. Hiermit kann ein Datenanbieter präzise die Art seiner Daten beschreiben. Der Datenanbieter wird dann nicht nur gefunden, wenn die Anwendung nach genau solchen Daten sucht, sondern auch wenn sie nach allgemeineren Konzepten sucht. Schließlich möchte ein Museum auch in der Liste der allgemeinen Veranstaltungsorte auftauchen und nicht nur in der Liste der Museen. Aktuelle Entwicklungen wie Google Local [Goog04] oder YellowMap [YeMa06] erlauben zwar die Verknüpfung von räumlichen Daten mit Konzepthierarchien, jedoch sind dort die Einträge nur durch einen Punkt repräsentiert, so dass dies nicht zur Verwaltung von Dienstgebieten eingesetzt werden kann. Zudem sind keine Details über die intern eingesetzten Verarbeitungsansätze öffentlich verfügbar.

Da zum Zeitpunkt der Implementierung des Verzeichnisdienstes für das Nexus-System kein existierendes System mit den benötigten Eigenschaften gefunden werden konnte, wurde eine eigene Lösung implementiert. Diese basiert auf einer Hauptspeicheranfrageverarbeitungskomponente wie sie in Kapitel 5.2 vorgestellt wird, die zusätzlich ihre Daten in einer Datenbank persistent speichert. Sie kommuniziert mit den Föderationskomponenten über das SOAP Protokoll. Für Anfragen und Antworten wurden eigene XML-basierte Schnittstellensprachen entwickelt, die jedoch hier nicht näher erläutert werden sollen. Diese sind in [BDG+04] dokumentiert.

3.5 Alternative Ansätze

Föderierte Anfrageverarbeitungs- und Datenintegrationssysteme sind schon seit vielen Jahren ein Thema in der Forschungslandschaft. Inzwischen gibt es in diesem Bereich auch einige kommerzielle Produkte. In diesem Teilkapitel sollen die bisherigen Ansätze auf ihre Eignung als Integrationsmiddleware im Nexus-System hin überprüft werden. Die Analyse erfolgt entlang der folgenden Kriterien [SHG+04]:

- **Dynamik**. Ist der Ansatz geeignet, um mit häufig wechselnden Konfigurationen von Datenanbietern zurecht zu kommen?
- **Schema**. Welche Art von Schema stellt der Ansatz den Anwendungen zur Verfügung. Dies bestimmt hauptsächlich die Ausdrucksmächtigkeit der Anfragen und die Erweiterbarkeit des Schemas.
- **Integrationskosten**. Der Aufwand, um einen neuen Datenanbieter ins System zu integrieren.
- **Integrationsmodus**. Art der Homogenität, die erreicht wird.

- **Systemgröße**. Größenordnung der Anzahl der Datenanbieter in typischen Systemen.
- **Ergebnisgröße**. Größenordnung der Größe typischer Ergebnismengen in Byte. Je nach Schema kann eine große Ergebnismenge aus sehr vielen kleinen Informationseinheiten bestehen oder aus nur wenigen großen Objektblöcken.

Kommerzielle Integrationsprodukte

Kommerzielle Integrationsprodukte wie Data Joiner von IBM [IBM00] oder Attunity Connect [Attu01] sowie Ansätze aus der Forschung wie TSIMMIS [GPQ+97] oder die EXIP Plattform [PaVa02] konzentrieren sich darauf, eine vorab bekannte und festgelegte Menge von Datenanbietern zu integrieren und ein integriertes Schema für diese Anbieter zu entwickeln. Ansätze im Gebiet der Schemaintegration [BLN86] sowie Softwaresysteme für diesen Zweck, beispielsweise CLIO [HMN+99, MHH+01] haben einen ähnlich beschränkten Fokus. Sie betrachten das Problem, eine festgelegte Menge von Quellschemata (und die zugehörigen Daten) in ein optimal berechnetes oder vorher festgelegtes Zielschema zu transformieren. Den Ansätzen fehlt die Flexibilität, um mit dynamisch wechselnden Anbietern umgehen zu können. Zudem werden die Daten der verschiedenen Anbieter nicht semantisch integriert, es besteht keine Möglichkeit Mehrfachrepräsentationen zu verschmelzen. Wenn ein neuer Datenanbieter hinzugefügt werden soll, erfordert dies typischerweise Anpassungen, die von einem Administrator durchgeführt werden müssen. Zudem führt das nachträgliche Einfügen eines Datenanbieters häufig zu Informationsverlust, wenn dessen Schema nicht zum feststehenden globalen Schema passt. Schemaintegrationssoftware kann jedoch lokal bei einem Datenanbieter eingesetzt werden, um dessen Daten in das globale Schema des Umgebungsmodells zu transformieren.

Peer-to-Peer Systeme

Weitere Datenintegrationsansätze basieren auf einer Peer-to-Peer-Architektur [BGK+02, HIST03]. Solche Systeme sind für eine hohe Dynamik ausgelegt und erlauben es ad hoc Datenanbieter hinzuzufügen und zu entfernen. Traditionelle Datenintegrationssysteme mit einem festgelegten, globalen Schema sind hier ungeeignet. Stattdessen werden hier die Datenanbieter (Peers) direkt gekoppelt, indem jeweils paarweise die lokalen Schemata über Domänenrelationen (engl. domain relations) und Koordinationsformeln (engl. coordination formulas) [BGK+02] oder Partnerabbildungen (engl. peer mappings) [HIST03] verknüpft werden. Eine Anwendung sendet Anfragen direkt an einen der Datenanbieter und verwendet dabei dessen lokales Schema. Der Datenanbieter greift dann auf die anderen Datenanbieter über die Verknüpfungen zu, wobei gleichzeitig Schema und Daten gemäß der Verknüpfung angepasst werden.

Um n Datenanbieter zu koppeln sind zwischen $O(n)$ und $O(n^2)$ Verknüpfungen notwendig. Wenn nur wenige Verknüpfungen zwischen den Datenanbietern etabliert werden hat dies den Vorteil, dass der Aufwand zum Einfügen eines neuen Datenanbieters gering ist. Dem steht der Nachteil gegenüber, dass die Daten viele Transformationsschritte durchlaufen müssen auf ihrem Weg vom speichernden Datenanbieter zur Anwendung. Jeder Transformationsschritt führt dabei mögli-

cherweise zu Informationsverlust. Wenn andererseits viele Verknüpfungen etabliert werden, genügt in vielen Fällen ein einziger Transformationsschritt. Dafür muss aber jeder Datenanbieter mit nahezu jedem anderen verknüpft werden. Entsprechende Schema- und Datentransformationen müssen abgeleitet werden, wobei dies bisher noch nicht vollautomatisch möglich ist.

Während ein globales Schema in dynamischen Umgebungen durchaus zu Einschränkungen führt, so überwiegen doch die Vorteile. Ein globales Schema bietet eine einheitliche Schnittstelle für Anwendungen und reduziert die Zahl der Transformationsschritte, die erstellt werden müssen und die bei der Verarbeitung von Anfragen ausgeführt werden müssen. Die Beschränkung auf eine konkrete Anwendungsdomäne vereinfacht zudem die Erstellung des globalen Schemas. Das Nexus-System ist trotzdem offen für zukünftige Erweiterungen, da es erlaubt, Erweiterungen zum Schema zu definieren, siehe [Nick05].

Information Manifold

Information Manifold [LRO96] verfolgt das Konzept, die Daten mehrerer Anbieter durch Verbunde zu kombinieren. Ähnlich wie bei dem in dieser Arbeit verfolgten Ansatz wird dort der Inhalt eines Anbieters durch Sichten über dem globalen Schema beschrieben. Dort werden ebenfalls eindeutige Identifikatoren verwendet, um mehrfache Repräsentationen des selben Realweltobjekts bei verschiedenen Anbietern zuzuordnen. Jedoch werden dort die Repräsentationen nicht semantisch verschmolzen. Für das dort verwendete globale Schema sind keine Erweiterungen vorgesehen. Bei der Verarbeitung von Anfragen werden alle Kombinationen von passenden Datenanbietern aufgezählt, und das Ergebnis von einem Anbieter dient als Eingabe für den nächsten Anbieter. Dieses Vorgehen folgt aus dem Verbund-basierten Ansatz. Es wird schnell ineffizient, wenn es viele relevante Anbieter zu einer Anfrage gibt, da es entweder viele Kombinationen gibt oder lange Verbundketten. Bei den in Kapitel 4 vorgestellten Verfahren können alle Datenanbieter gleichzeitig befragt werden und deren Ergebnisse werden anschließend in der Föderationskomponente integriert.

Semantic Mediation

Die Semantic Mediation [GLM02] oder SEAL [MSS+02] setzen einen Mediator ein, um die Verteilung der Daten auf viele Anbieter vor der Anwendung zu verbergen. Im Mediator können Abbildungen zwischen den Ontologien der Datenanbieter und der Ontologie des Mediators definiert werden. Die hier eingesetzten Ontologien enthalten sowohl Schema- als auch Instanzkonzepte, so dass diese nur in eng abgesteckten Anwendungsszenarien eine vernünftige Größe und Komplexität haben. Anfragen werden über logische Deduktionen aus der Ontologie heraus beantwortet, was im Allgemeinen für große Datenmengen ineffizient ist.

Semantic Web

Das World Wide Web und die Websuchmaschinen sind die derzeit am weitesten verbreiteten verteilten Informationssysteme. Diese enthalten derzeit weder Schema- noch Semantikinformationen, weshalb Anfragen auf reine Schlüsselwortsuchen

begrenzt sind. Da die Indexe der Suchmaschinen durch Web Crawler aktuell gehalten werden, wird das Hinzufügen oder Entfernen einer Datenquelle in der Regel erst mit einiger Verzögerung im Index nachvollzogen, so dass die Ergebnismengen ungültige Verweise enthalten können.

Das Semantic Web [Bern99] ist gedacht als die nächste Generation des World Wide Web und enthält auch maschinenlesbare Informationen zusätzlich zum Inhalt, der für menschliche Leser gedacht ist. Dies bietet Suchmaschinen die Möglichkeit, Informationen wesentlich intelligenter aufzubereiten als sie es bisher konnten. Technisch gesehen ist das Semantic Web eine Sammlung von portablen Standards, um Ontologien zu repräsentieren. Die Forschung in diesem Bereich konzentriert sich auf die Entwicklung von Sprachen mit denen Ontologien repräsentiert werden können, sowie auf Mechanismen zur Verarbeitung von Ontologien [FHH+00, HeGu00, W3C02]. Inferenzmaschinen ziehen Schlüsse über die Entitäten einer Ontologie. Sie helfen auch bei der Erstellung von Abbildungen zwischen Ontologien, indem sie Beziehungen zwischen den Entitäten zweier Ontologien ableiten.

Das Semantic Web kümmert sich nur um Metadaten. Im Gegensatz zum Nexus-System verarbeitet oder transformiert es keine Daten, und es strebt auch nicht an, eine Infrastruktur für die Integration großer Informationsmengen darzustellen. Es ermöglicht Schlüsse über alle möglichen Arten von Information zu ziehen, und kann somit als Basis für ein Integrationssystem dienen. Es befasst sich jedoch nicht mit dem Auffinden und Zusammenführen von Daten. Im Gegensatz dazu hat das Nexus-System ein sehr eingeschränktes Anwendungsfeld und strebt dabei an, die Daten vieler Anbieter zu integrieren und effizient zu verarbeiten.

Grid

Ein Grid [FoKe99] ist eine verteilte Datenverarbeitungsinfrastruktur, die sich aus vielen verschiedenen verteilten Ressourcen und Diensten zusammensetzt. Virtualisierungsdienste [RNC+02] erlauben es, den Anwendungen gegenüber verschiedene Transparenzstufen zu etablieren. Ein Verzeichnisdienst beispielsweise ermöglicht Lokationstransparenz, so dass Anwendungen damit konkrete Datenquellen und Dienste, die gegebene Anforderungen erfüllen, finden können, ohne diese namentlich vorher kennen zu müssen. Verteilungstransparenz, wie sie durch das Umgebungsmodell erreicht werden soll, wird im Grid Bereich jedoch derzeit nur als Ziel zukünftiger Forschungsarbeiten diskutiert [BGH+04].

In Data Grids arbeiten viele große und kleine Dienste, die über viele Organisationen verstreut sind, zusammen und stellen gemeinsam große verteilte Datenbestände bereit [HJK+01]. Die Hauptthemen der Forschungsarbeiten drehen sich um Replikation und Konsistenz. In den meisten Fällen liegen die Daten im Granulat von sehr großen Dateien vor. Anwendungen stellen keine komplexen Anfragen sondern benutzen das Grid, um die eine Datei zu finden, welche komplett die gewünschten Daten enthält. Im Gegensatz dazu soll im Nexus-System die Integration auf Objektebene erfolgen und die Daten vieler Anbieter werden zu einem globalen Umgebungsmodell zusammengefügt.

Der Verzeichnisdienst aus [Hosc02] speichert für jede Datenquelle einen Verweis zusammen mit weiteren Metadaten. Die gespeicherten Datenquellen können jedoch nur einzeln abgerufen werden, da die Antworten auf Anfragen immer nur einen

Verweis auf die passendste Datenquelle enthalten. Auch unterstützt der Verzeichnisdienst weder die Integration der Daten mehrerer Anbieter noch die Verschmelzung von Mehrfachrepräsentationen.

Compute Grids konzentrieren sich auf die Bearbeitung leicht partitionierbarer Probleme mit Hilfe generischer Ressourcen. In [SWG+03] wurde ein Grid-basiertes, verteiltes Anfrageverarbeitungssystem vorgestellt, das die Ausführung von Anfragen auf mehrere Knoten verteilt, und das dabei die Daten mehrerer Anbieter verknüpfen kann. Jedoch müssen hierbei die Datenquellen explizit in der Anfrage genannt werden, und die Verknüpfung der Daten erfolgt durch Funktionen, die in der Anfrage explizit vorgegeben werden. Lediglich die Ausführung der Anfrageverarbeitung wird dynamisch verteilt. Im Gegensatz dazu können im Nexus-System die Anwendungen deklarativ bestimmen, an welchen Daten sie interessiert sind, und die Daten verschiedener Anbieter werden von der Integrationsmiddleware zusammengefügt.

Zusammenfassung

Die Charakteristika der alternativen Architekturansätze werden in Tabelle 1 stichwortartig zusammengefasst. Jedes System kann einen Teil der an eine Integrationsmiddleware für kontextbezogene Anwendungen gestellten Anforderungen erfüllen, jedoch erfüllt kein System alle Anforderungen. Aus diesem Grund wurde die Integrationsmiddleware des Nexus-Systems von Grund auf neu entwickelt und dabei speziell auf die Anforderungen abgestimmt [SHG+04].

Tabelle 1. Charakteristika der alternativen Architekturansätze

Ansatz	Dynamik	Schema	Integrationskosten
Integrationsprodukte	niedrig	global	mittel
Peer-to-Peer Systeme	mittel - hoch	lokal	hoch
Information Manifold	mittel	global	mittel
Semantic Mediation	mittel	global, aber erweiterbar	mittel
WWW/Suchmaschinen	hoch	keines	niedrig
Semantic Web	hoch	globale Basis, lokale Verfeinerungen	niedrig
Grid	mittel	keines	niedrig
Nexus	mittel	global, aber erweiterbar	mittel

Ansatz	Integrationsmodus	Systemgröße	Ergebnisgröße
Integrationsprodukte	Schema, Global-als-Sicht	~10 Anbieter	~Megabyte
Peer-to-Peer Systeme	Schema, Partnerabbildungen	~10^2 Anbieter	~Megabyte
Information Manifold	Instanzen, Lokal-als-Sicht	~10^3 Anbieter	~Megabyte
Semantic Mediation	Instanzen, als Konzepte in der Ontologie	~10-10^2 Anb.	~Megabyte
WWW/Suchmaschinen	Zugriff, schlüsselwortbasierte Auswahl	~10^9 Anbieter	~Kilobyte
Semantic Web	Zugriff, konzeptbasierte Auswahl	~10^9 Anbieter	~Kilobyte
Grid	Zugriff, eigenschaftsbasierte Auswahl	~10^3 Anbieter	~Gigabyte
Nexus	Instanzen, Lokal-als-Sicht	~10^6 Anbieter	~Megabyte

Die Angaben zu Systemgröße und Ergebnisgröße in Tabelle 1 sind grobe Schätzungen.

3.6 Bewertung

In diesem Kapitel wurde die Architektur der Integrationsmiddleware im Nexus-System, und damit der dort verfolgte Föderationsansatz, vorgestellt. Das Nexus-System ist darauf ausgerichtet, flexibel und offen zu sein, weshalb die Integrationsmiddleware mit dynamisch wechselnden Konfigurationen von Datenanbietern zurecht kommen muss. Das Datenaustauschformat ist auf die Anforderungen des Datenmodells abgestimmt. Es unterstützt Mehrfachtypen und Mehrfachattribute. Die Anfragesprache ist auf die Anforderungen typischer kontextbezogener Anwendungen ausgerichtet. Einfache Selektions- und Projektionsanfragen sind hierbei ausreichend. Der Verzeichnisdienst nutzt die Spezifika typischer Anfragen aus, die meist die gesuchten Objekte nach Typ und Ort einschränken. Für jeden Datenanbieter werden Dienstgebiet und Typliste gespeichert. Anfragen werden mit der in Kapitel 5.2 beschriebenen Anfrageverarbeitungskomponente verarbeitet, da kein existierendes System gefunden werden konnte, das die geforderte Funktionalität bietet. Ebensowenig konnten adäquate alternative Architekturansätze ausfindig gemacht werden. Entweder kommen die Ansätze nicht mit der Dynamik der Datenanbieter, mit der Menge der zu verwaltenden Daten, mit räumlichen Daten oder mit der Notwendigkeit, Mehrfachrepräsentationen zu verschmelzen zurecht. Um die in Kapitel 2.2 aufgestellten Anforderungen der Autonomie, Integration und Transparenz erfüllen zu können, blieb nur die Entwicklung einer eigenen, speziell an diese Anforderungen angepassten Integrationsmiddleware.

4

Konzeption der föderierten Anfrageverarbeitung

In diesem Kapitel wird ein Konzept zur Verarbeitung von Anfragen in der Föderationskomponente erarbeitet und die Anfrageverarbeitung hinsichtlich der Interaktion mit anderen Komponenten des Nexus-Systems beleuchtet. Es werden wesentliche Interaktionsmuster aufgezeigt und es wird beschrieben, welche Daten bei einer Interaktion kommuniziert werden, und wie diese berechnet werden. Zunächst wird in Kapitel 4.1 die Semantik einer Anfrage formal definiert. Dann wird in Kapitel 4.2 ein Basisalgorithmus zur Verarbeitung von Gebietsanfragen vorgestellt. Anschließend werden in Kapitel 4.3 verschiedene Ansätze zur Verarbeitung von föderierten Nachbarschaftsanfragen diskutiert und experimentell verglichen. In Kapitel 4.4 wird untersucht, wie mehrfach repräsentierte Objekte in der Anfrageverarbeitung behandelt werden müssen. Es wird analysiert, unter welchen Umständen die Verteilung der Daten zu einer Entität in der realen Welt auf mehrere Datenanbieter zu unvollständigen oder fehlerhaften Anfrageergebnissen führt, und wie dies durch eine Anpassung der Anfragesemantik oder durch zusätzliche Informationen wie spezielle Relationenobjekte (sogenannte MRep-Relationenobjekte, siehe Kapitel 4.4.1.1) kompensiert werden kann. Schließlich werden weitere Maßnahmen zur Steigerung der Effizienz der föderierten Anfrageverarbeitung in Kapitel 4.5 präsentiert und die Umsetzung der föderierten Anfrageverarbeitung durch die vorgestellten Verfahren in Kapitel 4.6 bewertet.

4.1 Semantik einer Anfrage bei Mehrfachattributen

Die Semantik der Anfrage bestimmt, welche Objekte den *restriction* Teil der Anfrage erfüllen und somit in die Ergebnismenge dieser Anfrage gehören. Dies gilt gleichermaßen für Gebietsanfragen wie auch für Nachbarschaftsanfragen. Nachdem die formale Notation in Kapitel 4.1.1 eingeführt wurde, wird die Semantik der Anfragen in Kapitel 4.1.2 formal definiert. Die Anfragesemantik muss speziell darauf angepasst werden, dass Objekte mehrere Instanzen eines Attributs enthalten können.

4.1.1 Formale Notation

Zur Definition der Semantik einer Anfrage wird im nächsten Kapitel die Prädikatenlogik der ersten Stufe eingesetzt. Die erste Stufe ist hierbei ausreichend, da zwar keine konkreten Prädikate und Objektprädikate benannt werden, die Definitionen aber auch keine Quantoren darauf enthalten. Die Definitionen gelten jeweils für beliebige gültige Prädikate und Objektprädikate. Die Symbole, die in den Definitionen auftauchen, werden in Tabelle 2 eingeführt. Die Symbolik basiert hierbei auf den in [Nick05] eingeführten Symbolen zur Beschreibung von Objektinstanzen und erweitert diese um geeignete Definitionen für Prädikate, Objektprädikate und Selektionen.

Tabelle 2. Symbole der formalen Notation

Symbol	Bedeutung
o	Objekt
α	Attribut
υ	Wert eines Attributs
κ	Attributschema
A_{erf}	Erforderliche Attribute
A_{opt}	Optionale Attribute
dom	Wertebereich eines Attributtyps
O, R	Menge von Objekten, Ergebnismenge
$\mathfrak{B}$	Zahlenmenge der Wahrheitswerte
P	Prädikat
Q	Objektprädikat
σ	Selektion
$m \in M = \{E, A\}$	Existiert-Semantik oder Alle-Semantik
$n \in N = \{S, W\}$	Streng-Semantik oder Weich-Semantik

4.1.2 Formale Definition der Anfragesemantik

Die Semantik einer Anfrage wird von zwei orthogonalen Aspekten beeinflusst: der Verarbeitung von Mehrfachattributen und der Verarbeitung von fehlenden Attribu-

ten, welche vergleichbar sind mit NULL Werten in einer Datenbank. Bei der Verarbeitung von Mehrfachattributen kann ein Prädikat entweder genau dann als erfüllt gewertet werden, wenn mindestens eine Instanz des Attributs das Prädikat erfüllt (Existiert-Semantik), oder wenn alle Instanzen des Attributs das Prädikat erfüllen (Alle-Semantik). Das Ergebnis eines Prädikats, dessen Vergleichsattribut fehlt, wird analog zu [Codd86] mit einem besonderen Wahrheitswert *null* markiert. Die Verknüpfung mehrerer Prädikate in dieser dreiwertigen Logik erfolgt wie in [Codd86] beschrieben, siehe auch Abbildung 41 in Kapitel 4.4.3. Für jedes Objekt kann so zu einer gegebenen Anfrage ein Wahrheitswert berechnet werden, der angibt, ob das Objekt in der Ergebnismenge zu dieser Anfrage enthalten ist oder nicht. Objekte, deren Wahrheitswert *null* beträgt, können hierbei entweder zu der Ergebnismenge gehören (Weich-Semantik), oder von ihr ausgeschlossen sein (Streng-Semantik). Die Semantik einer Anfrage wird immer durch die Kombination der beiden Aspekte definiert.

Die Existiert-Streng-Semantik stellt die intuitive Semantik dar, da Objekte schon dann als Treffer gelten, wenn sie in irgendeiner Weise die Anfrage erfüllen, nicht jedoch, wenn die in der Anfrage verglichenen Attribute unbesetzt sind. Diese Semantik ist auch die voreingestellte Semantik, und wird immer dann angewandt, wenn nicht explizit eine andere Semantik gefordert wird.

Zur formellen Beschreibung der Semantiken werden folgende Definitionen eingeführt.

Definition 4.1: Objekt

$$o \subseteq \{(\alpha_i, \upsilon_i) \,|\, \alpha_i \in A_{erf} \cup A_{opt} \wedge \upsilon_i \textit{ ist vom Attributtyp } \kappa(\alpha_i)\}$$

Ein Objekt o ist eine Menge von Attribut-Wert-Paaren (α_i, υ_i) und besteht aus einer Teilmenge aller möglichen Attribut-Wert-Paare. Bei einem Attribut-Wert-Paar muss das Attribut α_i entweder in der Menge der erforderlichen Attribute A_{erf} oder in der Menge der optionalen Attribute A_{opt} enthalten sein. Der Wert υ_i muss im Wertebereich des zugehörigen Attributtyps $\kappa(\alpha_i)$ sein und den richtigen Basisdatentyp aufweisen. Ein Objekt kann mehrere Attribut-Wert-Paare zum selben Attribut haben. Dies wird als Mehrfachattribut bezeichnet. Weitere Details können in [Nick05] nachgelesen werden. Objekttypen und Attributschemata sind an dieser Stelle unerheblich.

Definition 4.2: Objektmenge

$$O \subseteq \{o_i\}$$

Eine Objektmenge O ist eine Teilmenge aller möglichen Objekte o_i. Wenn eine Objektmenge das Ergebnis einer Anfrage ist wird sie als Ergebnismenge R bezeichnet.

Definition 4.3: Prädikat

$$P_{\alpha',\upsilon'}: dom(\kappa(\alpha')) \rightarrow \mathfrak{B}$$

Ein Prädikat $P_{\alpha',\upsilon'}(\upsilon)$ auf einem Attribut α' ordnet einem Wert υ im Wertebereich des Attributs einen Wahrheitswert zu, der angibt, ob der Wert in der vom Prädikat geforderten Relation zum Vergleichswert υ' steht. Beispiele für Prädikate sind Gleichheitsvergleiche, Größer- oder Kleiner-Vergleiche, sowie räumliche Vergleiche wie räumliche Überlappungen (INTERSECTS) oder räumliches Enthaltensein (CONTAINS). Ein Attribut α', das in einem Prädikat einer Anfrage ausgewertet wird, wird im Kontext dieser Anfrage als Vergleichsattribut bezeichnet.

Definition 4.4: Objektprädikat

$$Q^{m,n}_{P_{\alpha',\upsilon'}}: \{o_i\} \rightarrow \mathfrak{B}$$

Ein Objektprädikat $Q^{m,n}_{P_{\alpha',\upsilon'}}$ ordnet einem Objekt o_i einen Wahrheitswert zu, der angibt, ob das Prädikat $P_{\alpha',\upsilon'}$ unter Berücksichtigung der Semantik m,n auf den Attribut-Wert-Paaren des Objekts zu wahr ausgewertet werden kann. Hierbei wählt $m \in M = \{E, A\}$ entweder die Existiert-Semantik oder die Alle-Semantik aus. $n \in N = \{S, W\}$ wählt entweder die Streng-Semantik oder die Weich-Semantik aus.

Definition 4.5: Komplementäre Semantik

$\overline{m} \in M \backslash m$ und $\overline{n} \in N \backslash n$ wählt die jeweils komplementäre Semantik aus.

Definition 4.6: Selektion

$$\sigma_{Q^{m,n}_{P_{\alpha',\upsilon'}}}(O) = R = \left\{ o \in O \,|\, Q^{m,n}_{P_{\alpha',\upsilon'}}(o) \right\}$$

Eine Selektion $\sigma_{Q^{m,n}_{P_{\alpha',\upsilon'}}}$ erzeugt aus einer gegebenen Objektmenge O eine neue Ergebnismenge R, die nur diejenigen Objekte enthält, bei denen das Objektprädikat $Q^{m,n}_{P_{\alpha',\upsilon'}}$ zu wahr ausgewertet werden kann.

Ausgehend von diesen Definitionen können nun die Objektprädikate zu den vier verschiedenen Anfragesemantiken wie folgt definiert werden.

Definition 4.7: Objektprädikat mit Existiert-Streng-Semantik

$$Q^{E,S}_{P_{\alpha',\upsilon'}}(o) = (\exists(\alpha,\upsilon) \in o{:}\alpha = \alpha') \wedge (\exists(\alpha,\upsilon) \in o, \alpha = \alpha'{:}P_{\alpha',\upsilon'}(\upsilon))$$

Ein Objekt o wird genau dann als Treffer gewertet, wenn es mindestens ein Attribut-Wert-Paar (α, υ) mit dem gesuchten Attribut α' enthält (keine NULL Werte erlaubt), und wenn auf mindestens einem Attribut-Wert-Paar mit dem gesuchten Attribut das gegebene Prädikat $P_{\alpha',\upsilon'}$ zu wahr ausgewertet werden kann.

Definition 4.8: Objektprädikat mit Existiert-Weich-Semantik

$$Q^{E,W}_{P_{\alpha',\upsilon'}}(o) = (\forall(\alpha,\upsilon) \in o{:}\alpha \neq \alpha') \vee (\exists(\alpha,\upsilon) \in o, \alpha = \alpha'{:}P_{\alpha',\upsilon'}(\upsilon))$$

Ein Objekt o wird genau dann als Treffer gewertet, wenn es entweder kein Attribut-Wert-Paar (α, υ) mit dem gesuchten Attribut α' gibt (was vergleichbar mit einem NULL Wert im gesuchten Attribut ist), oder wenn auf mindestens einem Attribut-Wert-Paar mit dem gesuchten Attribut das gegebene Prädikat $P_{\alpha',\upsilon'}$ zu wahr ausgewertet werden kann.

Definition 4.9: Objektprädikat mit Alle-Streng-Semantik

$$Q^{A,S}_{P_{\alpha',\upsilon'}}(o) = (\exists(\alpha,\upsilon) \in o{:}\alpha = \alpha') \wedge (\forall(\alpha,\upsilon) \in o, \alpha = \alpha'{:}P_{\alpha',\upsilon'}(\upsilon))$$

Ein Objekt o wird genau dann als Treffer gewertet, wenn es mindestens ein Attribut-Wert-Paar (α, υ) mit dem gesuchten Attribut α' enthält (keine NULL Werte erlaubt), und wenn auf allen Attribut-Wert-Paaren mit dem gesuchten Attribut das gegebene Prädikat $P_{\alpha',\upsilon'}$ zu wahr ausgewertet werden kann.

Definition 4.10: Objektprädikat mit Alle-Weich-Semantik

$$Q^{A,W}_{P_{\alpha',\upsilon'}}(o) = (\forall(\alpha,\upsilon) \in o{:}\alpha \neq \alpha') \vee (\forall(\alpha,\upsilon) \in o, \alpha = \alpha'{:}P_{\alpha',\upsilon'}(\upsilon))$$

Ein Objekt o wird genau dann als Treffer gewertet, wenn es entweder kein Attribut-Wert-Paar (α, υ) mit dem gesuchten Attribut α' gibt (was vergleichbar mit einem NULL Wert im gesuchten Attribut ist), oder wenn auf allen Attribut-Wert-Paaren mit dem gesuchten Attribut das gegebene Prädikat $P_{\alpha',\upsilon'}$ zu wahr ausgewertet werden kann.

In Abbildung 8 werden die Unterschiede zwischen den Anfragesemantiken veranschaulicht. Dafür wird für eine Beispielanfrage und deren negiertes Pendant gezeigt, welche Objekte bei einer bestimmten Anfragesemantik die Anfrage erfüllen und welche nicht.

Anfrage: Finde alle Museen, deren Thema Naturkunde ist.

Objekte				Erfüllt Anfrage in			
oid	Typ	Name	Thema	Existiert-Streng-Semantik	Existiert-Weich-Semantik	Alle-Streng-Semantik	Alle-Weich-Semantik
1	Museum	Rosenstein	Naturkunde Völkerkunde	ja	ja	nein	nein
2	Museum	Löwentor	Naturkunde	ja	ja	ja	ja
3	Museum	Linden	Völkerkunde	nein	nein	nein	nein
4	Museum	Wilhelma	NULL	nein	ja	nein	ja

Anfrage: Finde alle Museen, deren Thema **NICHT** Naturkunde ist.

Objekte				Erfüllt Anfrage in			
oid	Typ	Name	Thema	Existiert-Streng-Semantik	Existiert-Weich-Semantik	Alle-Streng-Semantik	Alle-Weich-Semantik
1	Museum	Rosenstein	Naturkunde Völkerkunde	ja	ja	nein	nein
2	Museum	Löwentor	Naturkunde	nein	nein	nein	nein
3	Museum	Linden	Völkerkunde	ja	ja	ja	ja
4	Museum	Wilhelma	NULL	nein	ja	nein	ja

Abbildung 8: Objektqualifikation bei verschiedenen Anfragesemantiken.

Weiterhin können jetzt die folgenden Rechenregeln abgeleitet werden, welche auch anschaulich am Beispiel in Abbildung 8 nachvollzogen werden können.

Satz 4.11: Prädikat-Komplementär-Eigenschaft

$$\forall \upsilon: \neg(P_{\alpha', \upsilon'}(\upsilon) \wedge \neg P_{\alpha', \upsilon'}(\upsilon)) \qquad \forall \upsilon: (P_{\alpha', \upsilon'}(\upsilon) \vee \neg P_{\alpha', \upsilon'}(\upsilon))$$

Für ein Prädikat P und dessen Negation $\neg P$ gilt, dass ein Wert nie beide gleichzeitig erfüllt, aber eines von beiden immer erfüllt ist.

Beweis 4.11

Trivial, da ein Prädikat P für gegebene α', υ' und υ entweder immer wahr oder immer falsch ist. □

Satz 4.12: Objektprädikat-Nicht-Komplementär-Eigenschaft

$$(\exists o: (Q^{m,n}_{P_{\alpha', \upsilon'}}(o) \wedge Q^{m,n}_{\neg P_{\alpha', \upsilon'}}(o))) \vee (\exists o: \neg(Q^{m,n}_{P_{\alpha', \upsilon'}}(o) \vee Q^{m,n}_{\neg P_{\alpha', \upsilon'}}(o)))$$

Bei einem Objektprädikat Q_P zum Prädikat P und dem Objektprädikat $Q_{\neg P}$ zum negierten Prädikat $\neg P$ kann es sehr wohl sein, dass ein Objekt keines von beiden Objektprädikaten oder beide gleichzeitig erfüllt. Für

eine Selektion σ bedeutet dies, dass mit einem Prädikat und dessen Negierung unter Beibehaltung der Anfragesemantik nicht die gesamte Objektmenge in der Ergebnismenge enthalten ist. Anders ausgedrückt bedeutet dies, dass die Negation des Prädikats alleine nicht zur inversen Selektion bzw. zum inversen Objektprädikat führt.

$$Q_{\neg P}^{m,n}(o) \neq \neg Q_{P}^{m,n}(o)$$

Beweis 4.12

Für die beiden Streng-Semantiken lässt sich zeigen, dass ein Objekt, bei dem das zu überprüfende Attribut fehlt, die rechte Seite von Satz 4.12 erfüllt. Sei o ein Objekt, bei dem gilt:

$$\forall(\alpha, \upsilon) \in o : \alpha \neq \alpha'$$

Eingesetzt in Definition 4.7 oder Definition 4.9 ergibt sich

$$Q_{P}^{m,S}(o) = falsch \qquad \text{und} \qquad Q_{\neg P}^{m,S}(o) = falsch$$

da der linke Teil der Definitionen nicht erfüllt werden kann. Damit wird die rechte Seite von Satz 4.12 erfüllt. □

Für die beiden Weich-Semantiken lässt sich zeigen, dass ein Objekt, bei dem das zu überprüfende Attribut fehlt, die linke Seite von Satz 4.12 erfüllt. Sei o ein solches Objekt. Damit ergibt sich aus Definition 4.8 oder Definition 4.10

$$Q_{P}^{m,W}(o) = wahr \qquad \text{und} \qquad Q_{\neg P}^{m,W}(o) = wahr$$

da der linke Teil der Definitionen immer erfüllt ist. Damit wird die linke Seite von Satz 4.12 erfüllt. □

Satz 4.13: Inverses Objektprädikat

$$\neg Q_{P}^{m,n}(o) = Q_{\neg P}^{\bar{m},\bar{n}}(o)$$

Das inverse Objektprädikat kann durch gleichzeitige Prädikat-Negation und Semantik-Inversion erreicht werden.

Beweis 4.13

Der Beweis soll hier exemplarisch für die Existiert-Streng-Semantik durchgeführt werden. Für die übrigen Semantiken funktioniert der Beweis analog. Aus Definition 4.7 folgt:

$$\begin{aligned}\neg Q^{E,S}_{P_{\alpha',\upsilon'}}(o) &= \neg((\exists(\alpha,\upsilon)\in o{:}\alpha=\alpha') \wedge (\exists(\alpha,\upsilon)\in o,\alpha=\alpha'{:}P_{\alpha',\upsilon'}(\upsilon)))\\ &= \neg(\exists(\alpha,\upsilon)\in o{:}\alpha=\alpha') \vee \neg(\exists(\alpha,\upsilon)\in o,\alpha=\alpha'{:}P_{\alpha',\upsilon'}(\upsilon))\\ &= (\forall(\alpha,\upsilon)\in o{:}\alpha\neq\alpha') \vee (\forall(\alpha,\upsilon)\in o,\alpha=\alpha'{:}\neg P_{\alpha',\upsilon'}(\upsilon))\\ &= Q^{A,W}_{\neg P_{\alpha',\upsilon'}}(o)\end{aligned}$$

□

Korollar 4.14: Inverse Selektion

$$\sigma_{\neg Q^{m,n}_{P_{\alpha',\upsilon'}}}(O) = O - \sigma_{Q^{m,n}_{P_{\alpha',\upsilon'}}}(O) = \sigma_{Q^{\bar{m},\bar{n}}_{\neg P_{\alpha',\upsilon'}}}(O)$$

Eine Selektion mit einem negierten Objektprädikat ist schlecht zu berechnen, da hierfür immer die Gesamtmenge O notwendig ist. Besser ist es da Satz 4.13 anzuwenden und die Semantik umzukehren sowie die Negation zum Prädikat zu verschieben, damit gleich konstruktiv die richtigen Objekte für die Ergebnismenge bestimmt werden können. Aus diesem Grund sollten auch bei Prädikaten, die durch boolesche Verknüpfungen verbunden sind, alle Negationen direkt zu den Prädikaten verschoben werden (per De Morganscher Gesetze).

Satz 4.15: Konjunktion und Disjunktion von Prädikaten

$$Q^{m,n}_{P1_{\alpha1',\upsilon1'}\wedge P2_{\alpha2',\upsilon2'}}(o) = Q^{m,n}_{P1_{\alpha1',\upsilon1'}}(o) \wedge Q^{m,n}_{P2_{\alpha2',\upsilon2'}}(o)$$

$$Q^{m,n}_{P1_{\alpha1',\upsilon1'}\vee P2_{\alpha2',\upsilon2'}}(o) = Q^{m,n}_{P1_{\alpha1',\upsilon1'}}(o) \vee Q^{m,n}_{P2_{\alpha2',\upsilon2'}}(o)$$

Ein Objektprädikat mit einer Konjunktion oder Disjunktion von Prädikaten kann durch eine Konjunktion oder Disjunktion von Objektprädikaten mit den entsprechenden einzelnen Prädikaten ersetzt werden.

Herleitung 4.15

Die Herleitung soll exemplarisch für die Disjunktion bei der Existiert-Weich-Semantik durchgeführt werden. Die Herleitung für die anderen Semantiken und für die Konjunktion funktioniert in gleicher Weise. Aus Definition 4.8 und Satz 4.15 folgt:

$$\begin{aligned}Q^{E,W}_{P1_{\alpha1',\upsilon1'}\wedge P2_{\alpha2',\upsilon2'}}(o) &= ((\forall(\alpha,\upsilon)\in o{:}\alpha\neq\alpha1') \vee (\exists(\alpha,\upsilon)\in o,\alpha=\alpha1'{:}P1_{\alpha1',\upsilon1'}(\upsilon)))\\ &\wedge ((\forall(\alpha,\upsilon)\in o{:}\alpha\neq\alpha2') \vee (\exists(\alpha,\upsilon)\in o,\alpha=\alpha2'{:}P2_{\alpha2',\upsilon2'}(\upsilon)))\\ &= Q^{E,W}_{P1_{\alpha1',\upsilon1'}}(o) \wedge Q^{E,W}_{P2_{\alpha2',\upsilon2'}}(o)\end{aligned}$$

Es wird für jedes Prädikat getrennt geprüft, ob das zugehörige Attribut fehlt oder ob eines der vorhandenen Attribute das Prädikat erfüllt. Es ist

ausdrücklich erlaubt, dass jedes Attribut unabhängig von den anderen geprüften Attributen vorhanden sein kann (nicht NULL) oder fehlen kann (NULL). Es müssen nicht alle Attribute gleichzeitig NULL oder nicht NULL sein.

Korollar 4.16: Konjunktion und Disjunktion von Objektprädikaten

$$\sigma_{Q^{m,n}_{P1 \wedge P2}}(O) = \sigma_{Q^{m,n}_{P1} \wedge Q^{m,n}_{P2}}(O) = \sigma_{Q^{m,n}_{P1}}(O) \cap \sigma_{Q^{m,n}_{P2}}(O)$$

$$\sigma_{Q^{m,n}_{P1 \vee P2}}(O) = \sigma_{Q^{m,n}_{P1} \vee Q^{m,n}_{P2}}(O) = \sigma_{Q^{m,n}_{P1}}(O) \cup \sigma_{Q^{m,n}_{P2}}(O)$$

Zum einen können Prädikate boolesch verknüpft werden und damit direkt eine Ergebnismenge aufgebaut werden. Zum anderen können die Prädikate einzeln ausgewertet werden und damit jeweils eine eigene Ergebnismenge aufgebaut werden. Die einzelnen Ergebnismengen müssen dann noch mit Mengenoperationen verknüpft und zu einer Gesamtergebnismenge zusammengefasst werden.

In diesem Kapitel wurde die Semantik von Anfragen auf Objekten mit Mehrfachattributen definiert. Es wurden die formalen Grundlagen gelegt, um Anfragen, die aus mehreren boolesch miteinander verknüpften Prädikaten bestehen, auf einzelne Selektionen abbilden zu können, deren Ergebnisse mit Mengenoperationen zusammengefügt werden können. Dies erlaubt die flexible Nutzung vorhandener Indexstrukturen. Besondere Aufmerksamkeit muss der Verarbeitung von Negationen gewidmet werden, welche eine Umkehr der Semantik erforderlich machen können. Die bisherige Diskussion geht davon aus, dass alle für eine Anfrage relevanten Objekte an einer zentralen Stelle gelagert werden.

4.2 Gebietsanfragen

Gebietsanfragen stellen den wichtigsten Anfragetyp im Nexus-System dar, mit dem Anwendungen sich die benötigten Informationen aus dem Umgebungsmodell beschaffen können. Sie zeichnen sich dadurch aus, dass die Anwendung das Gebiet vorgibt, in dem die Ergebnisobjekte liegen sollen. Typischerweise ist dies der Kartenausschnitt, den die Anwendung dem Benutzer gerade anzeigt. Es kann sich dabei aber auch um einen Korridor handeln, wie beispielsweise das Gebiet 500 Meter links und rechts der Autobahn, oder um einen Kreis, um etwa alle Hotels in einem Kilometer Umkreis anzuzeigen. Zusätzlich kann eine Gebietsanfrage weitere Vergleiche mit anderen Attributen enthalten, die in beliebiger Weise boolesch verknüpft sein können. In der Regel wird mindestens noch der Typ der gesuchten Objekte (Hotel, Museum, und so weiter) eingeschränkt. Häufig werden zusätzliche Eigenschaften gefordert, wie „finde alle Museen in Stuttgart, in denen eine Ausstellung zum Thema Naturkunde läuft“ oder „finde alle Drei-Sterne-Hotels im Umkreis von zwei Kilometern, in denen noch Nichtraucher-Doppelzimmer verfügbar sind“.

Durch die Konzeption der Architektur des Nexus-Systems (siehe Kapitel 3.1) können Gebietsanfragen auf relativ einfache Weise verarbeitet werden, wie in Abbildung 9 zu sehen ist. Hierbei werden zunächst Mehrfachrepräsentationen weit-

gehend außer Acht gelassen. Die notwendigen Anpassungen in der Anfrageverarbeitung zur Behandlung von Mehrfachrepräsentationen werden ausführlich in Kapitel 4.4 diskutiert.

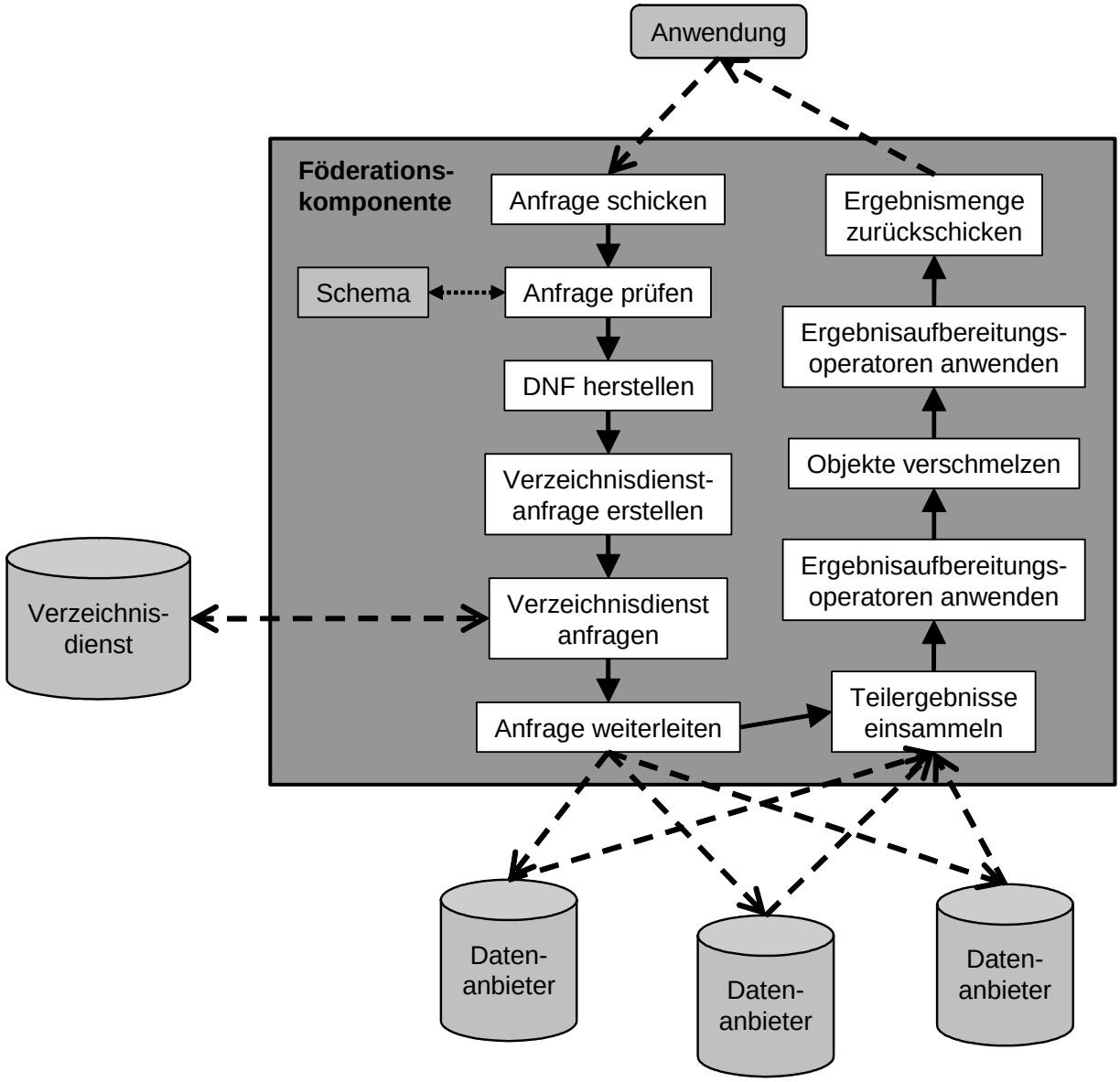


Abbildung 9: Verarbeitung von Gebietsanfragen in der Föderationskomponente

Der Ablauf der Verarbeitung gliedert sich wie folgt (siehe auch [NGS+01]).

1. **Anfrage schicken**. Die Anwendung schickt eine Gebietsanfrage an die Föderationskomponente.
2. **Anfrage prüfen**. Die Föderationskomponente prüft die Anfrage: Hat die Anfrage den korrekten Aufbau? Sind alle genannten erweiterten Schemata bekannt? Handelt es sich um eine Gebietsanfrage oder um eine Nachbarschaftsanfrage? Werden konkrete Datenanbieter vorgegeben, so dass direkt mit Schritt 6 fortgefahren werden kann?

3. **DNF herstellen (optional)**. Zur Vereinfachung der weiteren Schritte kann der *restriction* Teil einer Anfrage in die disjunktive Normalform (DNF) umgewandelt werden. Für die Verarbeitung von Mehrfachrepräsentationen (Kapitel 4.4) und für das Caching (Kapitel 5.5.1) ist diese Transformation zwingend erforderlich. Zur Herstellung der DNF werden zunächst mittels der De Morganschen Gesetze die Negationen zu den Literalen verlagert. Anschließend können die Distributivgesetze angewandt werden, bis der Term des *restriction* Teils die Form einer Disjunktion von Konjunktionstermen hat.

4. **Anfrage für Verzeichnisdienst erstellen**. Wie in Kapitel 3.4 erläutert, besteht die Anfrage an den Verzeichnisdienst aus einem Gebiet und aus einer Liste von Typen. Das Gebiet und die Liste wird erstellt, indem der *restriction* Teil der Anfrage traversiert wird und jeder Vergleichswert eines räumlichen Vergleichs mit dem bisherigen Gebiet vereinigt wird und jeder Vergleichswert eines Typvergleichs zur Liste der Typen hinzugefügt wird. Das initiale Gebiet und die initiale Liste sind leer. Diese einfache Vorgehensweise funktioniert hervorragend, solange keine unbeschränkten Negationen auftreten, die in der Anwendungsdomäne der kontextbezogenen Anwendungen aber auch nicht sinnvoll sind[1]. Bei den Gebieten werden Polygone mit Löchern unterstützt.

5. **Verzeichnisdienst anfragen**. Die soeben erstellte Verzeichnisdienstanfrage wird an selbigen geschickt. Dieser antwortet mit einer Liste der relevanten Datenanbieter. Das Dienstgebiet eines solchen Datenanbieters schneidet das Gebiet der Verzeichnisdienstanfrage, und der Datenanbieter speichert Objekte, deren Typ in der Typliste der Verzeichnisdienstanfrage auftaucht, oder deren Typ ein Subtyp der Typen in der Typliste ist.

6. **Anfrage weiterleiten**. Die Föderationskomponente schickt gleichzeitig an alle relevanten Datenanbieter eine Anfrage. Hierbei kann die ursprünglich von der Anwendung geschickte Anfrage direkt weitergeleitet werden. Der Datenanbieter ignoriert dann die Teile der Anfrage, für die er sich nicht zuständig fühlt. Der Verarbeitungsaufwand des Datenanbieters kann reduziert werden, wenn die weitergeleitete Anfrage speziell auf dessen Fähigkeiten und auf dessen gespeicherte Objekte angepasst wird. So können im *restriction* Teil in DNF diejenigen Konjunktionsterme weggelassen werden, die sich auf Typen beziehen, die dem Datenanbieter fremd sind. Die entsprechenden Projektionen im *filter* Teil können ebenfalls entfernt werden.

7. **Teilergebnisse einsammeln**. Die Antworten der relevanten Datenanbieter treffen nach und nach ein und werden zunächst getrennt gesammelt. Sobald die ersten Teilergebnismengen vollständig eingetroffen sind kann mit dem nächsten Schritt fortgefahren werden.

8. **Ergebnisaufbereitungsoperatoren anwenden**. Im ersten Ergebnisaufbereitungsschritt werden zum einen die Ergebnisaufbereitungsoperatoren

[1] Anfragen der Art „finde alle Objekte, die kein Museum sind“ können theoretisch als Scan über alle Objekte implementiert werden, was jedoch sehr aufwändig ist. Geschickter sind Anfragen der Art „finde alle Gebäude, die kein Museum sind“.

ausgeführt, die jedes Teilergebnis der Datenanbieter getrennt verarbeiten. Zum anderen werden verschmelzende Ergebnisaufbereitungsoperatoren ausgeführt, die Mehrfachrepräsentationen zusammenführen und eventuell zuvor auch erkennen. Ein Beispiel hierfür ist der in Kapitel 5.4.2.2 beschriebene Operator für sich überlappende Straßendatensätze.

9. **Objekte verschmelzen**. Im zweiten Ergebnisaufbereitungsschritt werden alle Objekte verschmolzen, welche die gleiche Objekt-ID besitzen. Hierfür werden die Objekte nacheinander von den Teilergebnismengen in eine Gesamtergebnismenge übertragen, welche zusätzlich eine Hash-Tabelle auf dem Objektidentifikator besitzt. Bei einer Mehrfachrepräsentation werden alle Attributinstanzen des hinzuzufügenden Objekts in das schon in der Gesamtergebnismenge befindliche Objekt kopiert. Nur bei exakt identischen Attributinstanzen, d. h. alle Attributteile der Instanz stimmen überein, wird nur eine Kopie gespeichert.
10. **Ergebnisaufbereitungsoperatoren anwenden**. Im dritten Ergebnisaufbereitungsschritt werden Ergebnisaufbereitungsoperatoren angewandt, welche auf der vereinigten Ergebnismenge arbeiten. Ein Beispiel hierfür ist der in Kapitel 5.4.2.2 beschriebene Generalisierungsoperator.
11. **Ergebnismenge zurückschicken**. Die vereinigte Ergebnismenge wird an die Anwendung als Antwort auf deren Anfrage zurückgesandt.

Es hat sich gezeigt, dass bei den vorgegebenen Anforderungen der Anwendungen die beschriebene hart verdrahtete Anfrageverarbeitung völlig ausreichend ist. Die Beschränkung der Anfragesprache auf Selektionen und Projektionen ermöglicht nur eingeschränkte Optimierungen (Reihenfolge der Prädikate, Verknüpfung mehrerer Prädikate auf Prädikat- oder auf Ergebnismengenebene).

4.3 Föderierte Nachbarschaftsanfragen

Verschiedenste Anwendungen nutzen Anfragen, um auf räumliche Daten zuzugreifen: ortsbezogene Anwendungen stellen nahegelegene interessante Orte (engl. points of interest) in einer Karte dar, räumliche Datenanalyseanwendungen suchen nach dem besten Standort für eine neue Filiale einer Ladenkette, Werkstücke in einer intelligenten Fabrik suchen die nächste verfügbare Maschine für ihre weitere Verarbeitung, und so weiter. In allen diesen Anwendungsgebieten wird auf die räumlichen Daten mit zwei verschiedenen Anfragetypen zugegriffen, mit Gebietsanfragen und mit Nachbarschaftsanfragen.

Gebietsanfragen setzen sich aus einem räumlichen Prädikat wie *intersects* (schneiden) oder *inside* (echt enthalten) und einem Anfragegebiet, das durch ein Rechteck oder ein Polygon gegeben ist, zusammen. Die Ergebnismenge einer Gebietsanfrage enthält all diejenigen Objekte, die das räumliche Prädikat erfüllen. Dieses Prädikat vergleicht das Anfragegebiet mit der Geometrie eines Objekts. Eine typische Gebietsanfrage soll zum Beispiel alle Objekte ermitteln, die innerhalb eines gegebenen Kartenausschnitts liegen.

Nachbarschaftsanfragen setzen sich aus einem Referenzpunkt und der Anzahl der zurückzuliefernden Objekte zusammen. Diese Anzahl wird gemeinhin mit k

bezeichnet. Die Ergebnismenge einer Nachbarschaftsanfrage enthält diejenigen k Objekte, die den geringsten Abstand zum Referenzpunkt aufweisen. Alle anderen Objekte sind mindestens genauso weit vom Referenzpunkt entfernt wie diese k Objekte. Eine typische Nachbarschaftsanfrage soll zum Beispiel die fünf nächstgelegenen Tankstellen ermitteln.

4.3.1 Motivation

Bislang wurden Nachbarschaftsanfragen nur an einzelne Datensätze gestellt, die zumeist lokal gespeichert waren, so dass eine Anfrageverarbeitungskomponente umfassenden Zugriff auf die Daten und auf die zugehörigen Indexstrukturen hatte. Im Umfeld des Nexus-Systems sind solche Nachbarschaftsanfragen nur eingeschränkt nützlich und nicht mit dem Ziel eines umfassenden Umgebungsmodells vereinbar, das sich nach außen hin als ein einziges großes Modell präsentiert, intern jedoch aus vielen einzelnen unabhängigen lokalen Umgebungsmodellen zusammengesetzt wird. Deshalb sollen hier nun Nachbarschaftsanfragen untersucht werden, die an eine lose gekoppelte Föderation aus *autonomen* Anbietern von räumlichen Daten gestellt werden. Eine Anfrageverarbeitungskomponente hat hier *keinen Einfluss* auf die Verteilung der Daten auf die einzelnen Datenanbieter. Die Ergebnismenge einer solchen Föderierten Nachbarschaftsanfrage (FNA) enthält die k nächstgelegenen Objekte aus dem vereinigten Datenbestand aller Datenanbieter. Der hier vorgestellte FNA-Algorithmus minimiert die Anzahl der an die Datenanbieter gestellten Anfragen und befragt dabei nur einen Bruchteil aller Datenanbieter. Des Weiteren ist der Algorithmus nicht auf exakte Objektzählungen oder Statistiken über die Objektdichte angewiesen. Dies ist äußerst wichtig, da es wegen der Autonomie der Datenanbieter sehr aufwändig und teuer ist solche Statistiken aktuell zu halten.

Die Vorteile von föderierten Nachbarschaftsanfragen lassen sich am Beispiel einer ortsbezogenen Restaurantsuche verdeutlichen. Viele Anbieter von räumlichen Daten speichern Informationen über Restaurants. Jede Kette von Schnellrestaurants betreibt ihre eigenen Server. Lokale Restaurants beauftragen einen externen Diensteanbieter, der einen logischen Server für jedes Restaurant aufsetzt. Elektronische Branchenbücher verschiedener Betreiber enthalten Informationen über Restaurants. Schließlich bieten Stadtmagazine und Gourmetführer jeweils Erfahrungsberichte und Bewertungen zu einzelnen Restaurants. Ein hungriger Benutzer möchte nicht erst einen der Datenanbieter auswählen, und dann auf dessen Angebot beschränkt sein. Er möchte auch nicht mühselig die Ergebnislisten verschiedener Datenanbieter vergleichen. Tatsächlich möchte der Benutzer mit einer einzigen Anfrage alle Datenanbieter gleichzeitig befragen und deren Ergebnisse in einer integrierten Ergebnisliste angezeigt bekommen. Genau dies können föderierte Nachbarschaftsanfragen leisten. Der hier vorgestellte Algorithmus minimiert dabei gleichzeitig die an die Datenanbieter gestellten Anfragen in Anzahl und Umfang, und erhöht somit erheblich die Leistungsfähigkeit des Gesamtsystems ohne dass der Benutzer eine nennenswerte Verlängerung der Antwortzeit bemerkt.

Der FNA-Algorithmus benutzt einen Verzeichnisdienst, der eine Liste aller verfügbaren Datenanbieter verwaltet, um für eine Anfrage die relevanten Datenanbieter zu ermitteln. Für jeden Datenanbieter speichert der Verzeichnisdienst dessen Namen, die Verbindungsparameter für dessen Anfrageschnittstelle sowie dessen

Dienstgebiet, welches die Geometrien aller dort gespeicherten Objekte einschließt. Ein solcher Verzeichnisdienst kann auf verschiedene Weise implementiert werden. Unter anderem kann hierfür ein OGC Catalogue Service [OGC99c, OGC02], ein FGDC clearinghouse [FGDC], oder sogar ein Verzeichnisdienst für Web Services [OGC03, PBJ03, UDDI00], der um Verarbeitungsfähigkeiten für räumliche Daten erweitert wurde, eingesetzt werden. Des Weiteren kann auch die in Kapitel 5.2 vorgestellte orts- und typbezogene Anfrageverarbeitungskomponente hierfür verwendet werden.

Der hier vorgestellte FNA-Algorithmus zielt auf Datenintegrationsszenarien, bei denen die verteilt vorliegenden Daten aus mindestens einem der folgenden Gründe nicht an einer zentralen Stelle repliziert und zusammengefasst werden können:

- Der integrierte Datenbestand ist zu groß.
- Die Datenanbieter agieren autonom und verbieten es, ihre Inhalte zu replizieren, z.B. um den Zugriff darauf beschränken und Zugriffsrechte überprüfen zu können.
- Der Datenbestand wird häufig aktualisiert, so dass es zu teuer wird, die vielen Repliken, die nötig sind, um viele gleichzeitige Nutzer bedienen zu können, aktuell zu halten.

In jedem der obigen Fälle kann eine Integrationsmiddleware, die eine Föderation der beteiligten Datenanbieter etabliert, Abhilfe schaffen.

4.3.1.1 Ziele und Beitrag

In diesem Kapitel werden neue Lösungen beschrieben für das Problem der Verarbeitung von Nachbarschaftsanfragen in einer Föderation von lose gekoppelten Anbietern von räumlichen Daten [SIG+04]. Es wird eine ganze Familie von Algorithmen detailliert beschrieben und analysiert. Die einzelnen Datenanbieter sind autonom. Die Integrationsmiddleware kann daher die Verteilung der Daten auf die Datenanbieter weder beeinflussen noch ändern. Es werden verschiedene Arten von Statistiken, die sich in ihrer Granularität unterscheiden, miteinander verglichen: keine Statistiken, ein grober Schätzwert der globalen Objektdichte sowie detaillierte Objektanzahlen pro Datenanbieter. Auf die Daten der Datenanbieter wird entweder per entfernten Nachbarschaftsanfragen oder per entfernten Gebietsanfragen zugegriffen. Entfernte Anfragen erfordern im Gegensatz zu lokalen Anfragen immer eine Kommunikation über das Netzwerk. Lokal vorhandene Indexe können nur von den Datenanbietern selbst bei der Verarbeitung der an sie gestellten Anfragen verwendet werden, nicht aber von der Integrationsmiddleware. Des Weiteren wird vorgeschlagen, die Anzahl der gleichzeitig angefragten Datenanbieter als Stellgröße einzusetzen, um entweder die Antwortzeit oder den Ressourcenverbrauch zu minimieren. Es werden vier Leistungskenngrößen eingeführt, um die Effizienz des Algorithmus zu bewerten: Antwortzeit, Aufwand, Anzahl der Iterationen und Kosten-Nutzen-Verhältnis. Es werden mehrere Experimente auf Zufallsdatensätzen durchgeführt, um die Skalierbarkeit des Algorithmus zu zeigen und um die verschiedenen Ansätze zu vergleichen. Es kann gezeigt werden, dass das Kosten-Nutzen-Verhältnis gegenüber dem besten bisher in der Literatur beschriebenen Ansatz um das bis zu 3,5-fache verbessert werden kann.

Der weitere Verlauf dieses Kapitels gliedert sich wie folgt. In Kapitel 4.3.2 wird das Problemfeld und die Systemumgebung des FNA-Algorithmus detailliert beschrieben und dieser in Bezug zu früheren Arbeiten zu Nachbarschaftsanfragen gesetzt. Die Hauptschleife des Algorithmus wird in Kapitel 4.3.3 erläutert. Dort werden auch die Hilfsfunktionen zum Berechnen des geschätzen Radius des Anfragebiets und zum Weiterleiten der Anfragen an die Datenanbieter vorgestellt. In Kapitel 4.3.4 werden die Randbedingungen und Parameter der Experimente beschrieben und deren Ergebnisse analysiert. Die Zusammenfassung in Kapitel 4.3.5 beschließt dieses Themengebiet.

4.3.2 Problemfeld

Das Problemfeld der Nachbarschaftsanfragen kann in vier Kategorien gegliedert werden. Im folgenden Abschnitt wird die Kategorie der föderierten Nachbarschaftsanfragen und das zugehörige Datenintegrationsszenario erläutert. Anschließend werden die weiteren Kategorien (lokale Nachbarschaftsanfragen, emulierte Nachbarschaftsanfragen sowie parallele und verteilte Nachbarschaftsanfragen auf gezielt aufgeteilte Datensätze) diskutiert und ihre Bedeutung für den FNA-Algorithmus erörtert. Die Erklärungen beschränken sich dabei auf zweidimensionale Punktobjekte. Die Algorithmen können aber leicht auf Objekte mit ausgedehnten und höherdimensionalen Geometrien verallgemeinert werden. In der Literatur werden zum Teil Verfahren diskutiert, die nur einen einzigen nächsten Nachbarn finden sollen. Eine besondere Behandlung dieses Sonderfalls ist hier jedoch nicht notwendig, und die weitere Diskussion konzentriert sich daher auf den allgemeinen Fall, bei dem eine vorher festgelegte Anzahl von *k* nächsten Nachbarn gefunden werden soll.

4.3.2.1 Föderierte Nachbarschaftsanfragen (FNA)

Föderierte Nachbarschaftsanfragen suchen in den Daten von vielen lose gekoppelten Datenanbietern nach den *k* nächsten Nachbarn zu einem gegebenen Referenzpunkt. Die Daten der Anbieter können sich dabei sowohl überlappen als auch ergänzen. Abbildung 10 zeigt ein Beispielszenario, in dem mehrere Datenanbieter ähnliche Objekte speichern, die z.B. Informationen über Restaurants enthalten.

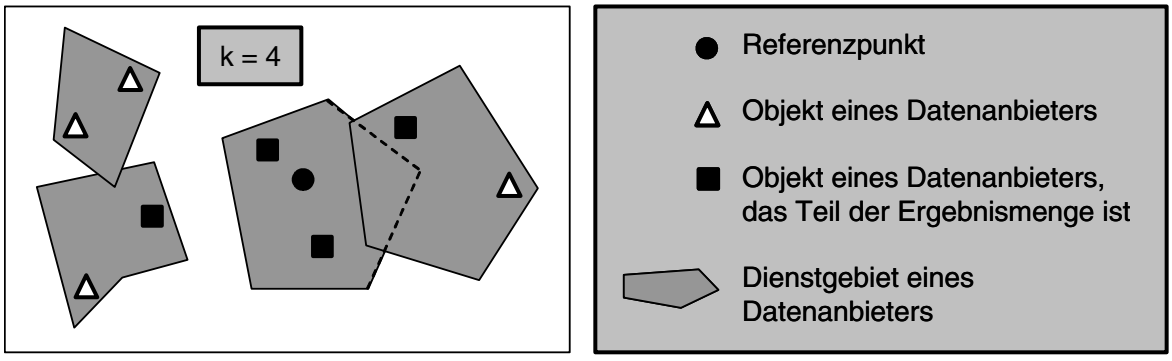


Abbildung 10: Restaurant-Objekte, die über verschiedene Datenanbieter verteilt sind (die Dienstgebiete der Datenanbieter können sich überlappen, ein Objekt gehört zu einem Datenanbieter, in dessen Dienstgebiet es sich befindet).

Ein Benutzer möchte in diesem Datenintegrationsszenario eine Liste der nächstgelegenen Restaurants erhalten, zu denen von irgendeinem der angeschlossenen Datenanbieter Informationen bereitgestellt werden. Der in Kapitel 4.3.3 vorgestellte FNA-Algorithmus erlaubt es, beim Benutzer den Eindruck zu erwecken, dass die Daten an einer zentralen Stelle gespeichert wären, obwohl sie tatsächlich bei den jeweiligen Datenanbietern verbleiben. Im Beispiel in Abbildung 10 sollen die vier nächstgelegenen Restaurants gefunden werden. Nur durch die Kombination der Daten mehrerer Anbieter können überhaupt genügend Ergebnisse gefunden werden. Dabei tragen manche Datenanbieter mit ihrem gesamten Datenbestand, manche mit einem Teil davon und manche gar nicht zur Ergebnismenge bei. Optimalerweise werden letztere erst gar nicht gefragt und die anderen nur nach so vielen Objekten, wie auch tatsächlich in der Ergebnismenge landen können.

Wie in Abbildung 11 zu sehen ist, setzt sich die Architektur eines Systems zur Verarbeitung von föderierten Nachbarschaftsanfragen aus drei Typen von Komponenten zusammen, welche einen Ausschnitt aus der in Kapitel 3.1 vorgestellten Architektur des Nexus-Systems bilden [NGS+01]: Datenanbieter mit räumlichen Daten, einem Verzeichnisdienst sowie einer Föderationskomponente in welcher der FNA-Algorithmus ausgeführt wird. Die Föderationskomponente bildet zusammen mit dem Verzeichnisdienst die Integrationsmiddleware. Zusätzlich gibt es noch die Anwendung, die hier aber neben dem Absetzen der initialen Anfrage keine weitere tragende Rolle spielt.

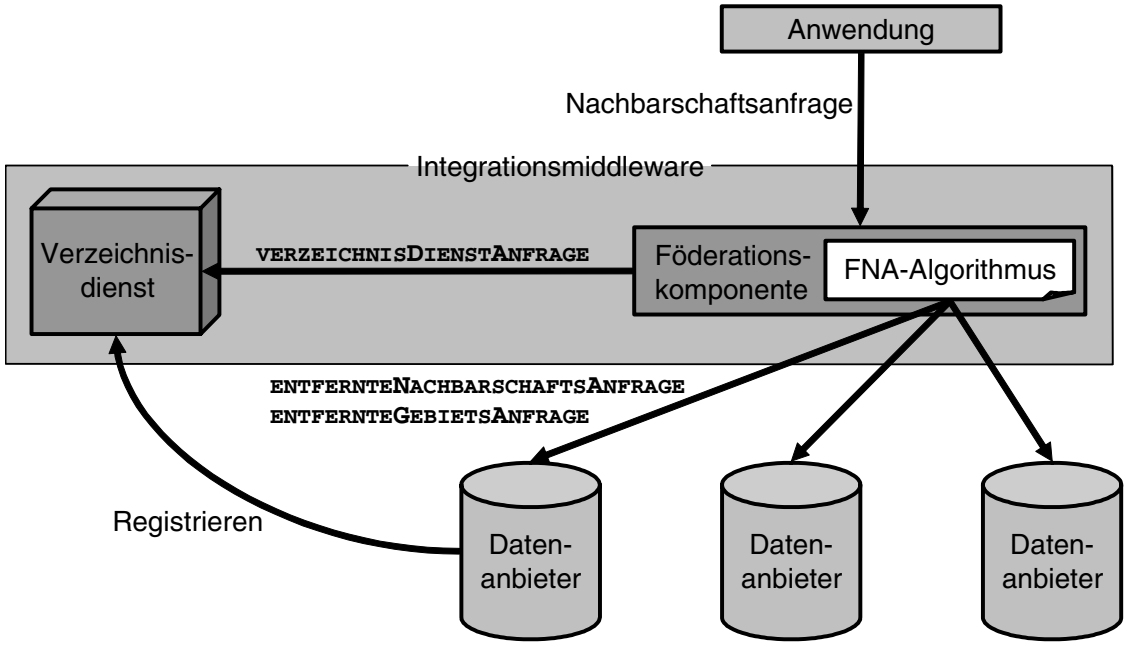


Abbildung 11: Systemarchitektur

Die Datenanbieter müssen mindestens eine der beiden folgenden Schnittstellen anbieten, damit die Integrationsmiddleware auf ihre räumlichen Daten zugreifen kann: eine Schnittstelle für entfernte Nachbarschaftsanfragen oder eine Schnittstelle für entfernte Gebietsanfragen. Bei der internen Organisation der Daten ist jeder Datenanbieter frei. Die Integrationsmiddleware hat keine Möglichkeit, etwaige lokale Indexstrukturen eines Datenanbieters auszunutzen. Ein Datenanbieter muss sich beim Verzeichnisdienst registrieren, um bei zukünftigen Anfragen berücksich-

tigt werden zu können. Hierfür muss er sein Dienstgebiet und die Verbindungsparameter, die notwendig sind, um auf seine Anfrageschnittstellen zugreifen zu können, an den Verzeichnisdienst übermitteln.

Das Dienstgebiet eines Datenanbieters umgibt die Geometrien aller dort gespeicherten Objekte. Die Form des Dienstgebiets wird durch eines der in der Simple Features Specification [OGC99a] definierten geometrischen Grundprimitive beschrieben. Typischerweise ist es ein Polygon, dessen Form sich an Verwaltungsgrenzen anlehnt, z.B. eines Stadtteils, einer Stadt oder eines Bundeslands. Das Dienstgebiet könnte auch der konvexen Hülle um alle Geometrien der gespeicherten Objekte entsprechen. In jedem Fall können sich die Dienstgebiete zweier Datenanbieter um mehrere Größenordnungen in ihrer Größe unterscheiden sowie sich gegenseitig überlappen oder enthalten. Kein Objekt eines Datenanbieters darf außerhalb von dessen Dienstgebiet liegen, da sonst das Objekt unter Umständen nicht gefunden werden würde.

Der Verzeichnisdienst wird von der Föderationskomponente benutzt, um zu einer Anfrage die jeweils relevanten Datenanbieter zu ermitteln. Hierfür speichert der Verzeichnisdienst die Dienstgebiete aller angemeldeten Datenanbieter. Der Verzeichnisdienst ist somit vergleichbar zu den inneren Knoten eines globalen räumlichen Indexes, bei dem die Datenanbieter dessen Blätter darstellen. Tatsächlich speichert der Verzeichnisdienst zu jedem Datenanbieter noch weitere Informationen wie z.B. die Typen der jeweils gespeicherten Objekte, siehe hierzu auch Kapitel 3.4. In der weiteren Diskussion des FNA-Algorithmus wird dies aber ohne Beschränkung der Allgemeinheit ausgeblendet. Hierfür wird angenommen, dass alle Datenanbieter Objekte des selben Typs speichern. Des Weiteren muss der Verzeichnisdienst Schätzwerte über die globale Objektdichte oder die Anzahl der Objekte, die bei jedem Datenanbieter gespeichert sind, speichern, je nach der eingesetzten Methode zur Bestimmung des initialen Radius des Anfragegebiets. Da sich die Dienstgebiete der Datenanbieter nur äußerst selten ändern, kann der Verzeichnisdienst leicht dupliziert werden, um dessen Skalierbarkeit zu gewährleisten und mehr Föderationsknoten bedienen zu können. Diese Vorgehensweise ist weniger praktikabel, wenn für jeden Datenanbieter auch die Anzahl der gespeicherten Objekte vorgehalten wird, da sich diese Anzahlen wesentlich häufiger als die Dienstgebiete ändern können. Der Verzeichnisdienst stellt eine Schnittstelle für Verzeichnisdienstanfragen bereit. Eine solche Anfrage besteht hier lediglich aus einem Anfragegebiet. Als Antwort wird eine Liste aller Datenanbieter zurückgeliefert, deren Dienstgebiet das gegebene Anfragegebiet schneidet (oder komplett darin enthalten ist).

Die Föderationskomponente führt den FNA-Algorithmus aus. Gegenüber den Anwendungen stellt sie eine Schnittstelle für einfache Nachbarschaftsanfragen [HjSa95, LLN02, RKV95] zur Verfügung. Die Föderationskomponente leitet Anfragen an die Datenanbieter weiter und integriert deren Antworten. Da die Föderationskomponente selbst keine Daten persistent speichert, kann sie leicht dupliziert werden, um mehr Benutzer gleichzeitig bedienen zu können. Auf diese Weise kann leicht die Skalierbarkeit der Föderationskomponente sichergestellt werden.

Um die Effizienz des FNA-Algorithmus zu bewerten werden zwei Optimierungsziele definiert: minimale Antwortzeit, die von der Anwendung wahrgenommen wird, sowie minimaler Ressourcenverbrauch, welcher sich aus dem Aufwand der Datenanbieter zur Beantwortung der an sie gestellten Anfragen und dem Aufwand

zur Übertragung dieser Ergebnismengen an die Föderationskomponente zusammensetzt. Der Aufwand ist ungefähr proportional zur Größe der Ergebnismenge und wird in Zeiteinheiten gemessen, siehe Kapitel 4.3.4. Die Transportkosten von der Föderationskomponente zur Anwendung sind bei allen Ansätzen gleich und können deshalb ausgeklammert werden. Während eine Anwendung so kurz wie möglich auf die Antwort einer Anfrage warten möchte (und somit die minimale Antwortzeit favorisiert), bevorzugt der Betreiber des Nexus-Systems den minimalen Ressourcenverbrauch, da dann mehr Benutzer gleichzeitig bedient werden können. Im weiteren Verlauf dieses Kapitels wird aufgezeigt, wie jedes dieser beiden Ziele für sich erreicht werden kann, und wie ein guter Kompromiss zwischen den beiden Extremen erreicht werden kann.

4.3.2.2 Lokale Nachbarschaftsanfragen

Lokale Nachbarschaftsanfragen zeichnen sich dadurch aus, dass bei ihrer Verarbeitung direkt auf die Daten und die zugehörigen Indexe zugegriffen werden kann. Bisherige Arbeiten in diesem Bereich haben sich einerseits darauf konzentriert, Algorithmen zur effizienten Traversierung der Knoten eines Indexbaums [ChFu98, HjSa95, HjSa99, RKV95] zu entwickeln, und dabei möglichst früh ganze Teilbäume zu überspringen. Häufig wurde hierbei der R*-Baum [BKSS90] als Indexstruktur betrachtet. Andererseits wurden spezielle auf Nachbarschaftsanfragen optimierte Indexstrukturen [Arya95, BEK+98, CNP99, YOTJ01] vorgeschlagen. In [LLN02] kann eine umfangreiche Diskussion über die bisherigen Ansätze nachgelesen werden. Alle diese Ansätze können nur zur lokalen Verarbeitung von Nachbarschaftsanfragen eingesetzt werden und nur, wenn lokal ein Index vorhanden ist. Die Ansätze können nicht erweitert werden auf den Zugriff auf entfernte Datenquellen, da zum einen die Datenquellen nicht die nötigen Zugriffsschnittstellen bieten, z.B. um auf den internen Index zuzugreifen, und da zum anderen die Ansätze von den geringen Kosten des Zugriffs auf lokale Daten ausgehen und die Verfahren völlig unangebracht sind, wenn die Zugriffskosten wegen des entfernten Zugriffs wesentlich höher ausfallen. Wie bereits in Kapitel 4.3.1 argumentiert wurde, stellt das Kopieren aller Daten auf eine Föderationskomponente, um dort lokal einen Index erstellen zu können, auch keine praktikable Lösung dar. Ein Datenanbieter kann jedoch die oben aufgeführten Verfahren einsetzen, um die an ihn gestellten Nachbarschaftsanfragen lokal zu verarbeiten.

4.3.2.3 Emulierte Nachbarschaftsanfragen

Emulierte Nachbarschaftsanfragen ermöglichen es, Nachbarschaftsanfragen an eine entfernte Datenquelle zu stellen, die nur eine Schnittstelle für entfernte Gebietsanfragen bereitstellt [LLN02]. Die Funktionalität einer Nachbarschaftsanfrage wird mittels einer Reihe von Gebietsanfragen emuliert, wobei deren Anzahl möglichst gering sein soll. Gleichzeitig soll auch die Zahl der übertragenen Objekte minimiert werden. Um dies zu erreichen, muss die Größe des Anfragegebiets der Gebietsanfrage möglichst passend geschätzt werden. In [LLN02] werden hierfür zwei Schätzmethoden vorgestellt, die beide auch im FNA-Algorithmus bei der Schätzung des Radius des Anfragegebiets zum Einsatz kommen, siehe Kapitel 4.3.3.3.

Der Ansatz aus [LLN02], der mit nur einer einzigen entfernten Datenquelle arbeitet, kann leicht auf eine föderierte Datenintegrationsumgebung anpasst werden: in jeder Iteration wird die Gebietsanfrage zunächst an den Verzeichnisdienst geschickt und anschließend an alle von dort zurückgelieferten relevanten Datenanbieter. Die hier untersuchten Varianten *EGA_ObjDichte* und *EGA_ObjAnzahl* repräsentieren das Ergebnis dieser Anpassung, siehe Kapitel 4.3.3.5, und werden in Kapitel 4.3.4 den Varianten, die entfernte Nachbarschaftsanfragen nutzen, gegenübergestellt. Im Gegensatz zu [LLN02], wo die Anfragegebiete rechteckig sind, können hier die Anfragegebiete beliebige regelmäßige n-Ecke sein, wodurch überschüssige Ergebnisobjekte in den Ecken des Anfragegebiets reduziert werden können. Zusätzlich werden hier Abkürzungsmechanismen eingeführt, so dass zu weit entfernte Datenanbieter übersprungen werden können und das Anfragegebiet verkleinert werden kann, sobald die Ergebnismenge k Objekte enthält, die jedoch noch nicht die letztendlich k nächstgelegenen sein müssen. Die Frage des parallelen Zugriffs auf mehrere Datenanbieter stellt sich in [LLN02] nicht, da es dort nur einen gibt.

4.3.2.4 Parallele und Verteilte Nachbarschaftsanfragen

Bisherige Arbeiten im Bereich der parallelen und verteilten Verarbeitung von Nachbarschaftsanfragen haben sich darauf konzentriert, wie die Daten am besten gezielt aufgeteilt werden können [BBB+97, FAA99, FAA01, FaBh93, LHY97, PAAA98]. Die Ansätze partitionieren den Datenraum und verteilen die Daten der Partitionen in cleverer Weise auf mehrere Festplatten, so dass bei der Verarbeitung einer gegebenen Anfrage möglichst viele Daten parallel von mehreren Festplatten gelesen werden können.

Wenn einzelne Festplatten als Datenanbieter aufgefasst werden, dann könnten eventuell obige Ansätze auf das hier diskutierte Problem übertragen werden. Jedoch hat hier die Integrationsmiddleware weder einen Einfluss auf die Verteilung der Daten auf die Datenanbieter noch auf die Form und Größe der Partition, die von einem Datenanbieter abgedeckt wird. Zusätzlich können sich die Partitionen bzw. Dienstgebiete der Datenanbieter überlappen. Zu einer gegebenen Anfrage enthält jeweils nur ein Bruchteil der Datenanbieter relevante Informationen, und Ziel der Untersuchungen hier ist es, möglichst wenige Anfragen an möglichst wenige Datenanbieter zu schicken, um eine Anfrage zu beantworten. Im Gegensatz zu den oben zitierten Ansätzen, welche immer auf alle Festplatten gleichzeitig zugreifen, wird hier vorgeschlagen, die Anzahl der gleichzeitig angefragten Datenanbieter als Stellgröße einzusetzen, um entweder die Antwortzeit oder den Ressourcenverbrauch zu minimieren, oder um einen Kompromiss dazwischen zu erzielen. Deshalb können die Ansätze zwar lokal bei einem Datenanbieter zur Beantwortung der an ihn gestellten Anfragen eingesetzt werden, nicht jedoch in der Föderationskomponente zur Verarbeitung von föderierten Nachbarschaftsanfragen.

In [PaMa96, PaMa01] werden mehrere Algorithmen vorgestellt, die versuchen, die Anzahl der anzufragenden Quelldatenbanken und die Anzahl der von jeder Datenbank angeforderten Objekte zu minimieren. Der dortige Ansatz basiert auf einem verteilten R-Baum, dessen oberer Teil auf einem Primärserver verwaltet wird und dessen Blattknoten bei den Quelldatenbanken verwaltet werden. Der Ansatz ist jedoch angewiesen auf Statistiken über die Anzahl der bei jeder Quelldatenbank gespeicherten Objekte, um die relevanten Quelldatenbanken zu bestimmen. Auch

werden die Quelldatenbanken entweder nur eine nach der anderen oder alle auf einmal angefragt. Die Experimente beschränken sich auf Systeme mit höchstens zehn Datenbanken. Größere Systeme, bei denen jede einzelne Datenbank nur einen kleinen Teil des gesamten Datenbestands abdeckt, werden nicht diskutiert. In den hier durchgeführten Experimenten (siehe Kapitel 4.3.4.2) wird dieser Ansatz durch die *ENA_ObjAnzahl* Variante repräsentiert, siehe Kapitel 4.3.3.3.

Eine ähnliche Systemarchitektur, die aus einem Primärserver und Quelldatenbanken besteht, wird in [KFK96] vorgeschlagen. Dort soll die Antwortzeit minimiert werden, indem die Anfrageverarbeitung soweit parallelisiert wird, wie es der Durchsatz des Netzwerks erlaubt. Hierfür wird die optimale Gruppengröße berechnet. Eine Gruppe besteht jeweils aus benachbarten Objekten, die bei derselben Quelldatenbank gespeichert werden. Dieser Ansatz ist hier jedoch nicht anwendbar, da die Integrationsmiddleware keinen Einfluss auf die Verteilung der Daten hat. Des Weiteren soll hier gerade vermieden werden, alle potentiell relevanten Datenbieter gleichzeitig zu fragen, was jedoch dieser Ansatz als Lösungsstrategie vorsieht.

In [PaMa98] werden die Knoten eines R-Baums auf mehrere Festplatten verteilt. Wenn man den dortigen Ansatz auf das hiesige Problem überträgt, dann hat der verteilte R-Baum hier nur zwei Ebenen: den Wurzelknoten, welcher dem Verzeichnisdienst entspricht, und die Blattknoten, welche den Datenanbietern entsprechen. Der Ansatz sieht vor, bei der Traversierung des R-Baums jeweils alle Kindknoten des aktuellen Knotens gleichzeitig zu laden. Folglich bietet dieser Ansatz bei der hier vorliegenden Problemstellung keinen Vorteil.

In [Fagi96] wird ein Szenario adressiert, bei dem die Daten eines Objekts selbst partitioniert und auf verschiedene Systeme verteilt sind. Der dort vorgestellte Ansatz ist bestrebt, die Anzahl der von jedem einzelnen System geholten Kandidaten zu minimieren, die benötigt werden, um die global gesehen ähnlichsten Objekte zu finden. Dieser Ansatz ist jedoch nicht auf das Szenario hier übertragbar, da sich hier die Ähnlichkeit bei den Nachbarschaftsanfragen rein auf den Abstand der Geometrie des Objekts zum Referenzpunkt bezieht, und da die Geometrie selbst nicht verteilt vorliegt.

Zusammengefasst lässt sich sagen, dass sich die bisherigen verwandten Arbeiten mit der gezielten Aufteilung der Daten beschäftigen, auf Statistiken über die Objektanzahlen angewiesen sind, auf alle Datenanbieter gleichzeitig zugreifen oder auf eine andere Systemarchitektur abzielen. Im Gegensatz dazu kann der hier vorgestellte FNA-Algorithmus die Verteilung der Objekte auf die Datenanbieter nicht beeinflussen, bevorzugt Statistiken über die globale Objektdichte, setzt auf eine angemessene Parallelität bei der Weiterleitung von Anfragen an die Datenanbieter und kann auch entfernte Gebietsanfragen einsetzen, falls ein Datenanbieter keine Schnittstelle für entfernte Nachbarschaftsanfragen bereitstellen sollte. Die Unterscheidungsmerkmale des FNA-Algorithmus werden in Tabelle 3 zusammengefasst.

Die Experimente in Kapitel 4.3.4 werden zeigen, dass die folgenden Parameter den größten Einfluss haben:

- Parallelität: Das Kosten-Nutzen-Verhältnis kann bis zum 2,9-fachen gesteigert werden.

Tabelle 3. Unterscheidungsmerkmale des FNA-Algorithmus gegenüber verwandten Ansätzen

Parameter	Verwandte Ansätze aus der Literatur	Vorgeschlagener FNA-Algorithmus
Verteilung der Daten	• Optimale Verteilung der Daten wird mit einem Aufteilungsalgorithmus ermittelt.	• Kein Einfluss auf die Verteilung der Daten, da die Datenanbieter autonom sind.
Zugriff auf die Daten	• Alle Objekte, die sich in einem Block auf der Festplatte befinden, der einem Eintrag im Index zugeordnet ist, werden von der Festplatte gelesen. • Entfernte Nachbarschaftsanfragen.	• Entfernte Nachbarschaftsanfragen. • Entfernte Gebietsanfragen.
Ermittlung der relevanten Datenanbieter	• Alle Datenanbieter werden als relevant eingestuft, da die Daten gezielt verteilt wurden, um maximale Parallelität beim Zugriff auf die Daten zu erreichen. • Mit Hilfe von Objektanzahlen, welche im vorliegenden Szenario nur mit großem Aufwand aktuell gehalten werden können, und Dienstgebieten wird die minimale Menge von Datenanbietern bestimmt, bei der garantiert ist, dass alle Ergebnisobjekte nur bei diesen Datenanbietern gesucht werden müssen.	• Mittels Dienstgebieten, um die Anzahl der angefragten Datenanbieter zu reduzieren. • Dienstgebiete und Schätzungen der globalen Objektdichte, um zusätzlich die Anzahl der Iterationen zu reduzieren.
Anzahl der Iterationen	• Eine: alle relevanten Datenanbieter werden in einem einzigen Schritt ermittelt.	• Mehrere: die Menge der relevanten Datenanbieter wird iterativ verfeinert.
Parallelität	• Maximal: Anfragen werden an alle relevanten Datenanbieter gleichzeitig weitergeleitet. • Keine: eine Anfrage wird erst dann an den nächsten relevanten Datenanbieter geschickt, wenn die Antwort der vorigen Anfrage vorliegt.	• Angemessen: Anfragen werden parallel an mehrere Datenanbieter, aber nicht gleichzeitig an alle relevanten Datenanbieter geschickt. Dies reduziert den Ressourcenverbrauch während trotzdem auch die Antwortzeit minimiert wird.

- Ermittlung der relevanten Datenanbieter: Das Kosten-Nutzen-Verhältnis kann bis zum 2,5-fachen gesteigert werden.
- Zugriff auf die Daten: Das Kosten-Nutzen-Verhältnis kann bis zum 2,0-fachen gesteigert werden.

4.3.3 Der FNA-Algorithmus

Der FNA-Algorithmus benötigt einen Referenzpunkt (`RefPunkt`) und die Anzahl der zurückzuliefernden Objekte (k) als Eingabeparameter. Die Antwort enthält genau die k Objekte aus den Datenbeständen aller verfügbaren Datenanbieter, die am nächsten zum gegebenen Referenzpunkt liegen. Durch eine geeignete Festlegung des Grads der Parallelität kann der FNA-Algorithmus so eingestellt werden, dass er einen optimalen Kompromiss zwischen Antwortzeit und Ressourcenverbrauch erzielt.

Der FNA-Algorithmus setzt die folgenden externen Funktionen ein:

- Die grundlegenden Geometriefunktionen DIST (Abstand), BUFFER (Puffer), CONTAINS (echtes Enthaltensein) und INTERSECTS (Überschneidung), welche in der Simple Features Specification definiert werden [OGC99a].
- Die Funktion VERZEICHNISDIENSTANFRAGE, welche eine Liste aller Datenanbieter liefert, deren Dienstgebiet das gegebene Anfragegebiet schneidet.

4.3.3.1 Die drei Phasen des FNA-Algorithmus

Abbildung 12 zeigt den Ablauf des FNA-Algorithmus für den Fall, dass keine weiteren Statistiken verfügbar sind. Der Algorithmus gliedert sich in die folgenden drei Phasen.

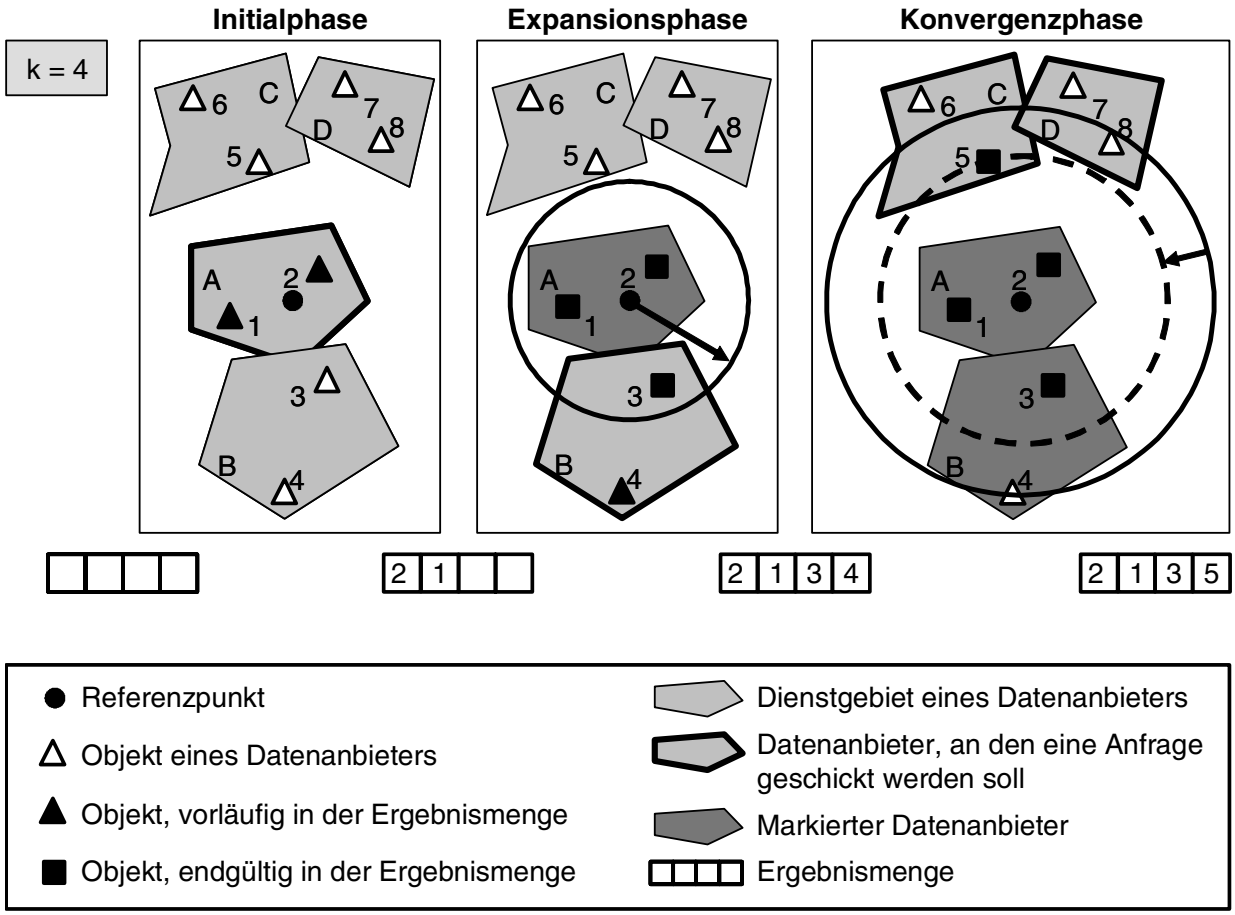


Abbildung 12: Die drei Phasen des FNA-Algorithmus ($k = 4$)

- Die **Initialphase** besteht aus der ersten Iteration der Hauptschleife des Algorithmus, siehe Abbildung 13. Der initiale Radius des Anfragegebiets wird geschätzt und Anfragen werden an alle Datenanbieter weitergeleitet, deren Dienstgebiet sich mit diesem Anfragegebiet überschneidet. In Kapitel 4.3.3.3 werden vier Ansätze diskutiert, wie der initiale Radius geschätzt werden kann. Im Beispiel in Abbildung 12 wird der initiale Radius auf null geschätzt, so dass das Anfragegebiet nur aus dem Referenzpunkt selbst besteht. Folglich wird zunächst nur der Datenanbieter A als relevant erachtet. Dieser liefert die Objekte 1 und 2 an die Föderationskomponente.

- In der **Expansionsphase** wird das Anfragegebiet iterativ so lange vergrößert, bis die Ergebnismenge *k* Objekte enthält. In jeder Iteration wird die Dichte der bislang gefundenen Objekte herangezogen, um den neuen Radius des Anfragegebiets zu schätzen. Dies wird in Kapitel 4.3.3.4 näher erläutert. Im mittleren Teil von Abbildung 12 wird das Anfragegebiet vergrößert. Nun wird Datenanbieter B befragt, welcher die Objekte 3 und 4 zurückliefert.
- Die **Konvergenzphase** wird benötigt, wenn schon *k* Objekte gefunden wurden, aber mindestens eines davon sich außerhalb des Anfragegebiets befindet, wie Objekt 4 in Abbildung 12. In dieser Phase wird nur noch nach Objekten gesucht, die näher beim Referenzpunkt liegen, als die bisher gefundenen Objekte. Hierfür wird der Radius des Anfragegebiets auf den Abstand des am weitesten entfernten Objekts in der vorläufigen Ergebnismenge gesetzt. Sobald ein nähergelegenes Objekt gefunden wird, kann der Radius verkleinert werden. Diese Phase stellt die letzte Iteration der Hauptschleife dar. In Abbildung 12 wird der Radius des Anfragegebiets auf den Abstand von Objekt 4 gesetzt. Nun sind auch die Datenanbieter C und D relevant, und liefern die Objekte 5 und 8 auf die an sie weitergeleiteten Anfragen. Objekt 5 ersetzt Objekt 4 in der Ergebnismenge, da es näher beim Referenzpunkt liegt. Eine weitere Iteration ist nicht mehr notwendig, da nun das am weitesten entfernte Objekt der Ergebnismenge innerhalb des Anfragegebiets liegt.

Von den drei Phasen muss nur die Initialphase ausgeführt werden. Die beiden anderen Phasen können unter bestimmten Bedingungen ausgelassen werden, siehe Kapitel 4.3.3.4.

4.3.3.2 Die Hauptschleife des FNA-Algorithmus

Abbildung 13 zeigt die Hauptschleife des FNA-Algorithmus in Pseudocode. Die Erklärungen zu den einzelnen Programmzeilen stehen als Kommentare zwischen den Zeilen. Die zusätzlichen im Algorithmus eingesetzten Funktionen werden in den nachfolgenden Unterkapiteln erläutert. Zur Vermeidung einer Endlosschleife enthält der Algorithmus eine Abbruchbedingung, welche mit den in Tabelle 4 dargestellten Konstanten konfiguriert wird.

Tabelle 4. Konstanten zur Konfiguration der Abbruchbedingung im FNA-Algorithmus

Konstante	Bedeutung
`GesamtzahlDerAnbieter`	Anzahl der beim Verzeichnisdienst registrierten Datenanbieter. Wenn alle abgehakt worden sind, kann die Hauptschleife abgebrochen werden.
`GesamtdienstgebietDerAnbieter`	Geometrische Vereinigung aller Dienstgebiete der beim Verzeichnisdienst registrierten Datenanbieter. Wenn dieses Gebiet komplett im Anfragegebiet enthalten ist, dann werden durch eine Vergrößerung des Anfragegebiets keine neuen Datenanbieter mehr gefunden und die Hauptschleife kann abgebrochen werden.
`MaximalErlaubteIterationen`	Obergrenze der Iterationen der Hauptschleife. Damit kann der Aufwand begrenzt werden, wobei unter Umständen dann nicht alle Ergebnisobjekte gefunden werden. Zehn Iterationen sollten in der Regel ausreichen.

```
    FÖDERIERTENACHBARSCHAFTSANFRAGE (RefPunkt, k) : ErgebnisSL
1   IterationsZähler := 0
2     // An diese Datenanbieter müssen keine weiteren Anfragen geschickt werden:
    AbgehakteAnbieterListe := new Liste()
3     // Sortierte Liste (SL) der Ergebnisobjekte, sortiert nach dem Abstand zum RefPunkt.
    ErgebnisSL := new SortierteListe(k)
4     // Radius des Anfragegebiets in der ersten Iteration schätzen:
    Radius := SCHÄTZEINITIALENRADIUS(RefPunkt, k)
5   do
6     IterationsZähler := IterationsZähler + 1
7     // Das Anfragegebiet ist ein regelmäßiges n-Eck um den RefPunkt mit dem gegebenen Radius.
      AnfrageGebiet := RefPunkt.BUFFER(Radius)
8       // Schicke eine VERZEICHNISDIENSTANFRAGE los, um die Datenanbieter zu ermitteln, deren
        // Dienstgebiet das Anfragegebiet schneidet. Entferne davon alle Datenanbieter, die
        // schon abgehakt sind. Erzeuge eine sortierte Liste (SL) der übrigen Datenanbieter.
        // Die Datenanbieter sind dabei nach dem Abstand ihres Dienstgebiets zum RefPunkt sortiert.
      AnbieterSL := ERSTELLEANBIETERSL(AnfrageGebiet,
                                       AbgehakteAnbieterListe, RefPunkt)
9       // Schicke eine Anfrage an alle Anbieter in der sortierten Liste und füge die zurückgelieferten
        // Objekte in die globale sortierte Liste der Ergebnisobjekte ErgebnisSL ein. Eventuell
        // können hier Abkürzungsmechanismen greifen, siehe Kapitel 4.3.3.5.
      BEFRAGEANBIETER(RefPunkt, k, AnfrageGebiet, AnbieterSL,
                      ErgebnisSL, AbgehakteAnbieterListe)
10      // Abbruchbedingungen, um eine Endlosschleife zu vermeiden, wenn zu wenige
        // Ergebnisobjekte gefunden werden.
      if ((AbgehakteAnbieterListe.size >= GesamtzahlDerAnbieter)
        or (AnfrageGebiet.CONTAINS(GesamtdienstgebietDerAnbieter))
          or (IterationsZähler >= MaximalErlaubteIterationen))
11      break
12    endif
13    AlterRadius := Radius
14      // Radius des in der nächsten Iteration zu verwendenden Anfragegebiets schätzen.
      Radius := SCHÄTZENÄCHSTENRADIUS(RefPunkt, k, ErgebnisSL,
                                      AlterRadius)
15  // Die Hauptschleife wird wiederholt solange noch nicht genügend Objekte gefunden wurden
    // oder solange der Radius des Anfragegebiets größer wird.
    while ((ErgebnisSL.size < k) or (Radius > AlterRadius))
16  return ErgebnisSL
```

Abbildung 13: Hauptschleife des Algorithmus für föderierte Nachbarschaftsanfragen

Der FNA-Algorithmus initialisiert als erstes den Iterationszähler, die Liste der Datenanbieter, an die keine weiteren Anfragen mehr weitergeleitet werden müssen, sowie die sortierte Liste der Ergebnisobjekte, welche darin nach ihrer Entfernung zum Referenzpunkt sortiert werden (Zeilen 1-3). Anschließend wird der Radius des in der ersten Iteration der Hauptschleife verwendeten Anfragegebiets ermittelt (Zeile 4), siehe Kapitel 4.3.3.3. Zu Beginn der Hauptschleife (Zeilen 6 und 7) des

FNA-Algorithmus wird der Iterationszähler erhöht und ein regelmäßiges n-Eck berechnet, das den Referenzpunkt zum Mittelpunkt hat und dessen Kanten einen Abstand gleich dem Radius des Anfragegebiets vom Mittelpunkt haben. Auf diese Weise wird das eigentlich kreisförmige Anfragegebiet durch ein Polygon angenähert, da in der Simple Features Specification [OGC99a] kein Kreis als geometrisches Grundprimitiv vorgesehen ist. Anschließend werden alle Datenanbieter ermittelt, deren Dienstgebiet das Anfragegebiet schneidet. Hiervon werden diejenigen entfernt, die nicht mehr gefragt werden müssen. Die verbleibenden Anbieter werden nach der Entfernung ihres Dienstgebiets zum Referenzpunkt sortiert (Zeile 8). An die verbliebenen Anbieter werden nun Anfragen geschickt, um die Objekte zu holen, die potentiell in der Ergebnismenge enthalten sein könnten (Zeile 9). Hierbei wird der Grad der Parallelität als Stellgröße eingesetzt, um den Algorithmus auf verschiedene Optimierungsziele anzupassen. Je nach Typ der Anfrage, die an die Datenanbieter weitergeleitet wird, können hier auch verschiedene Abkürzungsmechanismen zum Einsatz kommen, welche in Kapitel 4.3.3.5 näher erläutert werden. Die Abbruchbedingungen in Zeile 10 werden benötigt, um sicherzustellen, dass der Algorithmus auch dann terminiert, wenn weniger als k Objekte im gesamten Datenbestand enthalten sind. Dies wird insbesondere dann relevant, wenn die qualifizierenden Objekte der Nachbarschaftssuche durch weitere Prädikate eingeschränkt werden, oder wenn mehrere Datenanbieter eine Repräsentation desselben Objekts vorhalten. Siehe hierzu auch Kapitel 4.4. Der FNA-Algorithmus bricht ab, wenn alle Datenanbieter abgehakt wurden, wenn das Anfragegebiet die Vereinigung der Dienstgebiete aller Datenanbieter vollständig enthält, oder wenn eine vorher festgelegte Obergrenze der Anzahl der Iterationen erreicht wird. Am Ende der Hauptschleife wird der Radius des Anfragegebiets der nächsten Iteration berechnet (Zeile 14), siehe Kapitel 4.3.3.4. Die Hauptschleife kann beendet werden, wenn genügend Objekte gefunden wurden und wenn gleichzeitig das Anfragegebiet der nächsten Iteration nicht größer ist als das Anfragegebiet der vergangenen Iteration (Zeile 15). Nur dann sind die bisher gefundenen Objekte auch tatsächlich die k nächstgelegenen Objekte und können nicht mehr durch Objekte noch nicht befragter Datenanbieter verdrängt werden. Schließlich werden die gefundenen Objekte nach ihrer Entfernung zum Referenzpunkt sortiert an die anfragende Anwendung übermittelt (Zeile 16).

4.3.3.3 Schätzung des Radius des initialen Anfragegebiets (`SCHÄTZEINITIALENRADIUS`)

Das initiale Anfragegebiet wird in der ersten Iteration der Hauptschleife des FNA-Algorithmus verwendet. Es werden vier verschiedene Methoden vorgeschlagen, den Radius dieses initialen Anfragegebiets zu bestimmen. Diese Methoden werden in den Experimenten miteinander verglichen. Die folgenden zwei Methoden kommen ganz ohne Zusatzinformationen aus:

- **Null**. Das Anfragegebiet der ersten Iteration besteht nur aus dem Referenzpunkt selbst. Der Radius des initialen Anfragegebiets ist also null. Dies kann dazu führen, dass in der ersten Iteration gar keine relevanten Datenanbieter gefunden werden, wenn der Referenzpunkt außerhalb aller Dienstgebiete liegt. In diesem Fall wird in der zweiten Iteration ein heuristisch bestimmter Radius verwendet. Für die typische Größe und Verteilung von Datenanbietern in der Domäne der ortsbezogenen Anwendungen, siehe Experimente in Kapitel 4.3.4, hat sich ein Radius von einem Kilometer bewährt.

- **Maximum**. Der Radius des initialen Anfragegebiets wird auf unendlich gesetzt. Dies garantiert, dass in der ersten Iteration alle Datenanbieter als relevant angesehen werden und entsprechend an alle eine Anfrage weitergeleitet wird. Die meisten der in Kapitel 4.3.2.4 diskutierten Ansätze benutzen diese Methode, da sie die größtmögliche Parallelität bei der Befragung der Datenanbieter erlaubt, und dadurch vermeintlich zur kürzesten Antwortzeit führt. Im vorliegenden Szenario sind jedoch die Datenanbieter in den von ihnen verwalteten Daten und in ihrer Leistungsfähigkeit sehr heterogen. Darum ist mit dieser Methode auch garantiert, dass auf jeden Fall der langsamste Datenanbieter gefragt wird, was sehr häufig gar nicht nötig wäre. Zusätzlich führt diese Methode zum größtmöglichen Ressourcenverbrauch.

Die anderen beiden Methoden nutzen zusätzliche Metadaten über die bei den Datenanbietern gespeicherten Daten:

- **Objektdichte**. Es wird ein globaler Schätzwert der Dichte aller Objekte im Datenbestand herangezogen, um den Radius so zu schätzen, dass statistisch gesehen sich genau k Objekte im Anfragegebiet befinden. Das Verhältnis der Anzahl aller Objekte im gesamten Datenbestand zur Fläche des vereinigten Dienstgebiets aller Datenanbieter ist dabei gleich dem Verhältnis der Anzahl k der gesuchten Objekte zur Fläche des Anfragegebiets. Diese Methode funktioniert am besten, wenn die Objekte gleichmäßig in der Ebene verteilt sind. Je nachdem, wie gut diese Annahme zutrifft, kann mehr als eine Iteration der Hauptschleife des FNA-Algorithmus notwendig werden, bis die endgültige Ergebnismenge feststeht. In [LLN02] wird diese Methode die „Density-Based Method“ genannt.
- **Objektanzahl**. Von jedem Datenanbieter muss dessen Dienstgebiet und die exakte Anzahl der dort verwalteten Objekte bekannt sein. Damit kann das kleinste Anfragegebiet berechnet werden, das garantiert, dass schon in der ersten Iteration der Hauptschleife des FNA-Algorithmus alle Objekte der endgültigen Ergebnismenge gefunden werden. Hierfür werden die Datenanbieter nach der Entfernung des am weitesten vom Referenzpunkt entfernten Punkts ihres Dienstgebiets sortiert. Ein Anfragegebiet mit dem Radius des am weitesten entfernten Punkts des Dienstgebiets eines Datenanbieters garantiert, dass alle Objekte dieses Datenanbieters gefunden werden, und vor allem, dass alle anderen Datenanbieter als relevant erachtet werden, die potentiell noch näher gelegene Objekte verwalten könnten. Diese Liste der Datenanbieter wird beginnend mit der kleinsten Entfernung in aufsteigender Reihenfolge der Entfernungen abgearbeitet. Es werden so lange die Anzahlen der bei den entsprechenden Datenanbietern gespeicherten Objekte aufsummiert, bis die gesuchte Anzahl k erreicht oder überschritten wird. Die größte Entfernung des Dienstgebiets des zuletzt betrachteten Datenanbieters ergibt den Radius des initialen Anfragegebiets. Diese Methode garantiert gleichermaßen wie die *Maximum* Methode, dass nur eine Iteration benötigt wird und erachtet dabei wesentlich weniger Datenanbieter als relevant. Trotzdem führt auch diese Methode zu einem sehr hohen Ressourcenverbrauch, da das initiale Anfragegebiet meist viel zu groß geschätzt wird. Zusätzlich müssen die Objektanzahlen immer aktuell gehalten werden, um die Garantie, nach einer Iteration fertig zu sein, aufrecht erhalten zu können, was sehr teuer ist. In [LLN02] wird diese Methode die „Bucked-Based Method“ genannt, in [PaMa98] mit „threshold distance“ bezeichnet, und in [PaMa96] wird

dieser Radius in der „Parallel Nearest Neighbor Finding Method“ eingesetzt. Nirgends wird jedoch wie hier die Parallelität beschränkt, siehe Kapitel 4.3.3.5.

Eine erste Einschätzung legt nahe, dass im vorliegenden Szenario nur die *Null* und die *ObjDichte* Methode vernünftig einsetzbar sind. Die *Max* Methode ist ungeeignet wegen der viel zu hohen Anzahl der Datenanbieter, die als relevant eingestuft werden. Die *ObjAnzahl* Methode hat das Problem, dass sie von jedem Datenanbieter verlangt, dass er die Anzahl der bei ihm gespeicherten Objekte veröffentlicht und dass er diese Anzahl auch fortwährend aktualisiert. Die Voraussetzungen und Charakteristika der vier hier vorgestellten Methoden werden in Tabelle 5 zusammengefasst.

Tabelle 5. Methoden, um den Radius des initialen Anfragegebiets abzuschätzen

Methode	Abkürzung	Voraussetzungen	Charakteristika
Null	Null	Keine	Findet unter Umständen gar keine Datenanbieter in der ersten Iteration
Maximum	Max	Keine	Benötigt nur eine Iteration, betrachtet dabei aber viel zu viele Datenanbieter
Objektdichte	ObjDichte	Schätzwert der globalen Objektdichte	Findet meist nicht alle für das Endergebnis relevanten Datenanbieter in der ersten Iteration
Objektanzahl	ObjAnzahl	Anzahl der bei jedem Datenanbieter gespeicherten Objekte bekannt	Benötigt nur eine Iteration, betrachtet dabei wesentlich weniger Datenanbieter als die Max-Methode, es sind aber immer noch sehr viele

4.3.3.4 Schätzung des Radius der nächsten Iteration (SCHÄTZENÄCHSTENRADIUS)

In der Expansionsphase wird die Dichte der bisher gefundenen Objekte herangezogen, um den Radius des Anfragegebiets der nächsten Iteration zu berechnen. Das Verhältnis der Fläche des bisherigen Anfragegebiets zur Anzahl der bisher gefundenen Objekte ist dabei gleich dem Verhältnis der Fläche des neuen Anfragegebiets zur Anzahl k der gesuchten Objekte. Auf diese Weise passt sich das Anfragegebiet schnell der tatsächlichen Verteilung der für die Anfrage relevanten Objekte an, selbst wenn die Statistiken oder bisherigen Schätzungen der Objektdichte veraltet sind. Wenn noch keine Objekte gefunden wurden, wird der bisherige Radius verdoppelt. Wenn der bisherige Radius gleich null ist wird ein fester Wert verwendet, der heuristisch bestimmt wurde. Gute Ergebnisse konnten hierbei mit einem Radius von einem Kilometer erzielt werden.

Die Expansionsphase ist zu Ende sobald k Objekte gefunden wurden. Sie wird übersprungen, wenn schon in der Initialphase k Objekte gefunden wurden. Wenn das k-te Objekt, welches das am weitesten vom Referenzpunkt entfernte Objekt in der Ergebnismenge ist, innerhalb des in der letzten Iteration verwendeten Anfragegebiets liegt, dann kann die Konvergenzphase übersprungen werden. In diesem Fall würde in der nächsten Iteration ein kleineres Anfragegebiet eingesetzt werden, womit keine weiteren bisher noch unentdeckten Datenanbieter gefunden werden würden. Andernfalls wird die Konvergenzphase benötigt, und die Hauptschleife des FNA-Algorithmus wird ein letztes Mal ausgeführt. Das Anfragegebiet hat dabei einen Radius, welcher der Entfernung des k-ten Objekts in der vorläufigen Ergebnismenge entspricht.

4.3.3.5 Weiterleitung der Anfragen an die relevanten Datenanbieter (BEFRAGEANBIETER)

Je nachdem, welche Schnittstelle ein Datenanbieter bereitstellt, können Anfragen an ihn entweder als entfernte Nachbarschaftsanfragen oder als entfernte Gebietsanfragen gestellt werden. Im Folgenden werden zwei alternative Algorithmen vorgestellt, die jeweils einen der beiden Anfragetypen einsetzen, um die BEFRAGEANBIETER Funktion zu implementieren. Anschließend wird erläutert, wie die Parallelisierung der Anfragen funktioniert, und wie der Grad der Parallelität beschränkt werden kann. Schließlich wird argumentiert, warum es nicht sinnvoll ist, in der Manier eines Datenbank-Cursors die Objekte in der Reihenfolge ihres Abstands zum Referenzpunkt einzeln bei den Datenanbietern abzugreifen.

Die beiden Algorithmen zum Weiterleiten der Anfragen an die relevanten Datenanbieter setzen voraus, dass die Liste der anzufragenden Anbieter (`AnbieterSL`) aufsteigend nach dem Abstand ihres Dienstgebiets zum Referenzpunkt sortiert ist. Der Abstand bezieht sich dabei auf den nächstgelegenen Punkt des Dienstgebiets. Die sortierte Liste der Ergebnisobjekte (`ErgebnisSL`) ist in ihrer Größe auf k Objekte beschränkt. Wenn mehr als k Objekte in die Liste eingefügt werden, so fallen die am weitesten entfernten Objekte automatisch heraus.

Algorithmus mit entfernten Nachbarschaftsanfragen (ENA)

In der BEFRAGEANBIETER Funktion wird an jeden Anbieter in der `AnbieterSL` Liste eine entfernte Nachbarschaftsanfrage geschickt, wobei für jeden Anbieter die Anzahl der gesuchten Objekte individuell berechnet wird. Dieser individuelle Anfrageparameter wird `AnbieterSpezifischesK` genannt und gibt an, mit wie vielen Objekten dieser Datenanbieter höchstens noch zur Ergebnismenge beitragen kann. `AnbieterSpezifischesK` wird berechnet, indem von der Anzahl der gesuchten Objekte k die Anzahl derjenigen Objekte in der vorläufigen Ergebnismenge abgezogen wird, die näher am Referenzpunkt liegen als das Dienstgebiet dieses Anbieters. Dieser Abkürzungsmechanismus verringert die Anzahl der von einem Datenanbieter angefragten Objekte erheblich und verringert zur gleichen Zeit auch die Anzahl der Objekte, die später wieder verworfen werden müssen, weil sie durch näher liegende Objekte aus der Ergebnismenge verdrängt werden. Wenn `AnbieterSpezifischesK` gleich null ist, dann kann ein Datenanbieter gar keine Objekte zur Ergebnismenge beitragen. Somit muss auch keine Anfrage an diesen Datenanbieter geschickt werden. Andernfalls wird an den Datenanbieter eine entfernte Nachbarschaftsanfrage nach den `AnbieterSpezifischesK` nächsten Objekten geschickt, welche dann zur globalen Ergebnismenge `ErgebnisSL` hinzugefügt werden. Anschließend wird der Datenanbieter markiert, indem er zur Liste der abgehakten Anbieter `AbgehakteAnbieterListe` hinzugefügt wird, da an jeden Datenanbieter nur eine Anfrage geschickt werden muss. Die Datenanbieter werden in der Reihenfolge ihrer Sortierung in der `AnbieterSL` betrachtet. Dies führt dazu, dass `AnbieterSpezifischesK` von Anbieter zu Anbieter monoton schrumpft, da die Dienstgebiete der Anbieter immer weiter entfernt liegen, und da gleichzeitig immer mehr Objekte gefunden werden, die nahe beim Referenzpunkt liegen.

Algorithmus mit entfernten Gebietsanfragen (EGA)

Falls die Datenanbieter keine entfernten Nachbarschaftsanfragen unterstützen, so kann dies, ähnlich wie in [LLN02], kompensiert werden, indem stattdessen auf entfernte Gebietsanfragen zurückgegriffen wird. An jeden Anbieter in `AnbieterSL` wird eine Gebietsanfrage mit dem aktuellen Anfragegebiet geschickt und die zurückgelieferten Objekte werden zur globalen Ergebnismenge `ErgebnisSL` hinzugefügt. Falls die Ergebnismenge schon k Objekte enthält kann ein Abkürzungsmechanismus eingesetzt werden, der für jeden Datenanbieter das Anfragegebiet individuell anpasst. Dieses Anfragegebiet wird aus dem Radius `AnbieterSpezifischerRadius` abgegeleitet, welcher der Entfernung des aktuell k-ten Objekts in der Ergebnismenge entspricht. Ein Datenanbieter kann übersprungen werden, wenn der Abstand seines Dienstgebiets größer als `AnbieterSpezifischerRadius` ist. Ein Datenanbieter kann abgehakt werden (zur Liste `AbgehakteAnbieterListe` hinzugefügt werden), wenn sein Dienstgebiet komplett im Anfragegebiet enthalten ist. Dadurch, dass die Datenanbieter nach der Reihenfolge in `AnbieterSL` betrachtet werden, kann die Zahl der überspringbaren Datenanbieter maximiert werden.

Dieser Ansatz hat zwei Nachteile. Zum einen kann es notwendig sein, dass ein Datenanbieter mehrmals befragt wird, bis sein Dienstgebiet vollständig im Anfragegebiet enthalten ist und er abgehakt werden kann. Zum anderen kann ein Datenanbieter viel zu viele Objekte zurückliefern, da bei Gebietsanfragen nur das Anfragegebiet vorgegeben werden kann, aber keine Höchstzahl der Ergebnisobjekte. Beide Aspekte kommen besonders bei einer unregelmäßigen Verteilung der Objekte zum Tragen, wenn die Schätzung des Anfragegebiets missglückt.

Mehrere Anfragen parallel losschicken

Die BEFRAGEANBIETER Funktion kann leicht parallelisiert werden. Hierfür wird ein Vorrat an Threads angelegt, wobei in jedem Thread die ursprüngliche BEFRAGEANBIETER Funktion abläuft. Jeder Thread holt sich den nächsten noch unbearbeiteten Anbieter aus `AnbieterSL` und wendet gegebenenfalls die zuvor beschriebenen Abkürzungsmechanismen (`AnbieterSpezifischesK` und `AnbieterSpezifischerRadius`) an. Alle Threads starten gleichzeitig. Ein Thread fährt mit der Bearbeitung des nächsten Datenanbieters fort, nachdem das Ergebnis des zuvor von diesem Thread bearbeiteten Anbieters vorliegt. Je mehr Threads zur Verfügung stehen, desto kürzer ist die Zeit, bis das endgültige Ergebnis vorliegt, aber desto geringer ist auch die Effizienz der Abkürzungsmechanismen, so dass weniger Datenanbieter übersprungen werden können und mehr Objekte übertragen werden. Dies liegt darin begründet, dass weniger Antworten von Datenanbietern in den Abkürzungsmechanismen berücksichtigt werden können, da die zugehörigen Anfragen gerade erst parallel ausgeführt werden. Folglich führt dies zu einem höheren Ressourcenverbrauch und mehr Objekte werden unnötigerweise übertragen. Mutmaßlich ist die Antwortzeit am geringsten, wenn alle Datenanbieter gleichzeitig gefragt werden, wenn also die Threadanzahl gleich der Anzahl der relevanten Datenanbieter ist. Andererseits ist der Ressourcenverbrauch am geringsten, wenn die Datenanbieter einer nach dem anderen befragt werden, wenn also nur ein Thread in der BEFRAGEANBIETER Funktion verwendet wird. In Kapitel 4.3.4.2 wird beschrieben, wie ein guter Kompromiss erreicht werden kann.

Objekte einzeln bei den Datenanbietern abholen

Auf den ersten Blick erscheint die folgende Vorgehensweise in der BEFRAGEANBIETER Funktion die naheliegendste zu sein. Ähnlich wie beim Mischen-Schritt im Mergesort-Verfahren wird von jedem Anbieter zunächst ein Objekt geholt. Von diesen Objekten wird das mit dem geringsten Abstand zum Referenzpunkt in die Ergebnismenge eingefügt. Vom zugehörigen Anbieter wird jetzt das nächste Objekt geholt, um die freiwerdende Lücke zu füllen. Dies wird fortgesetzt, bis die Ergebnismenge k Objekte enthält, oder bis von keinem Datenanbieter mehr weitere Objekte geholt werden können. Auf diese Weise wird die Anzahl der übertragenen Objekte effektiv auf das absolute Minimum gedrückt. Es muss jedoch bedacht werden, dass jede Anfrage nach einem einzelnen nächsten Objekt ähnlich viel Kommunikationsaufwand erzeugt und zu ähnlich langen Latenzzeiten führt wie eine Anfrage nach k Objekten. Die Messungen in Kapitel 4.3.4 zeigen, dass die Verzögerungen durch die Kommunikation einen wesentlichen Anteil an der Gesamtantwortzeit stellen. Daher führt dieser Ansatz schon bei sehr kleinen Werten für k zu übermäßig vielen Interaktionen zwischen der Föderationskomponente und den Datenanbietern, wodurch die Antwortzeit erheblich verlängert wird. Deshalb wird dieser Ansatz nicht weiter betrachtet.

4.3.4 Experimente

In den Experimenten wird die Leistung des FNA-Algorithmus unter verschiedenen Bedingungen untersucht. Es werden zum einen die Anzahl k der gesuchten Objekte und zum anderen die Anzahl der in der BEFRAGEANBIETER Funktion verwendeten Threads variiert. Die Messungen wurden mittels eines Simulationsansatzes durchgeführt, bei dem der FNA-Algorithmus unverändert abläuft, aber bei dem alle entfernten Komponenten durch lokale Komponenten ersetzt wurden, die diese simulieren. Dieser Simulationsansatz ist notwendig, um aussagekräftige Messungen mit realistischen Mengen von Datenanbietern machen zu können, da nicht genügend reale Server zur Verfügung standen. Für jeden Messpunkt wurde der Mittelwert aus 1000 Anfragen mit zufällig gewählten Referenzpunkten berechnet. Die Referenzpunkte waren gleichmäßig im Universum verteilt. Die weiteren Randbedingungen der Experimente sind in Tabelle 6 verzeichnet.

Leider gibt es keinen frei verfügbaren Datensatz, der die in den Messungen hier benötigen Charakteristika aufweist. Die verfügbaren Datensätze sind entweder zu klein, oder ihre Daten sind nicht in überlappende Partitionen aufgeteilt, die den Dienstgebieten der Datenanbieter entsprechen könnten. Die TIGER/Line Datensätze der amerikanischen Behörde für Bevölkerungsstatistik (US Census Bureau, [UCB03]) sind zwar groß genug, zumindest wenn mehrere einzelne Datensätze zusammengenommen werden. Die Daten sind jedoch in disjunkte Gebiete (die Regierungsbezirke) aufgeteilt, so dass trotzdem eine künstliche Aufteilung herbeigeführt werden müsste. Insbesondere sollen sich die Dienstgebiete der Anbieter überlappen, so dass an einigen Stellen mehrere Anbieter zuständig sind. Des Weiteren sollten sich die Dienstgebiete erheblich in ihrer Größe unterscheiden, was bei den Regierungsbezirken nicht gegeben ist.

Die Charakteristika des Problemfelds können am besten nachgebildet werden, wenn die simulierten Datenanbieter nach speziellen Vorgaben zufällig generierte

Tabelle 6. Charakteristische Randbedingungen der Experimente

	Mindest-wert	**Höchst-wert**	**Durch-schnitt**
Universum	Deutschland (ca. 878 mal 610 km)		
Anzahl der Datenanbieter			10.000
Fläche der Dienstgebiete (Der Logarithmus der Fläche ist gleichverteilt)	101,0 m^2	225,3 km^2	75,5 km^2
Anzahl der Datenanbieter mit sich an der selben Stelle überlappenden Dienstgebieten	0	10	2,0
Abdeckung des Universums durch die Dienstgebiete			74,34%
Pro Anfrage Anteil an der Antwortzeit eines Datenanbieters (Fest pro Datenanbieter, wird zufällig zugewiesen mit einer negativen Exponentialverteilung)	10 ms	1000 ms	100,0 ms
Pro Ergebnisobjekt Anteil an der Antwortzeit eines Datenanbieters (Fest pro Datenanbieter, wird zufällig zugewiesen mit einer negativen Exponentialverteilung)	0,3 ms	10 ms	1,0 ms
Anzahl der Objekte (Die Positionen der Objekte sind gleichverteilt im Universum)			1.000.000
Anzahl der von einem Datenanbieter verwalteten Objekte	20	452	100,0
Anzahl der gesuchten nächsten Nachbarn (*k*)	1	1024	--
Anzahl der Threads	1-32, 25%-100%, 1+log, 2*log		

Datensätze verwalten. Hierfür wird für jeden Datenanbieter ein zufälliges Polygon als dessen Dienstgebiet generiert. Die Flächen der Dienstgebiete reichen dabei von der Fläche eines Ladengeschäfts bis zur Fläche einer mittelgroßen Stadt oder eines Stadtteils einer Großstadt. Die genauen Werte können Tabelle 6 entnommen werden. Die logarithmische Gleichverteilung ist notwendig, damit es ähnlich viele Dienstgebiete in jeder Größenklasse gibt. Ohne den Logarithmus würde es zu viele große Dienstgebiete geben, da der Mittelwert der Fläche bei der Hälfte des Maximalwerts läge. Die Positionen der Objekte sind gleichverteilt im Universum. Jedes Objekt wird zufällig einem der Datenanbieter zugeordnet, in dessen Dienstgebiet es sich befindet. Falls für ein Objekt kein passender Datenanbieter gefunden werden kann, wird es verworfen und ein neues Objekt generiert.

Die Antwortzeit eines simulierten Datenanbieters wird berechnet, indem der Pro-Anfrage-Anteil eines Datenanbieters und für jedes zurückgelieferte Objekt der Pro-Ergebnisobjekt-Anteil aufsummiert werden. Während der Pro-Anfrage-Anteil die typischen Kommunikationsverzögerungen und Latenzen im Internet repräsentiert, trägt der Pro-Objekt-Anteil dem lokalen Verarbeitungsaufwand und den mengenbezogenen Transportkosten Rechnung. In [PaMa96] wird experimentell bestätigt, dass diese lineare Formel eine sinnvolle Annäherung darstellt. Die beiden Anteile der Antwortzeit sind jeweils negativ exponentiell verteilt, so dass es viele schnelle Datenanbieter gibt, aber eben auch einige sehr langsame. Dies entspricht dem

Ergebnis von Untersuchungen im Web und bei E-Commerce Anwendungen, die gezeigt haben, dass ein nicht zu vernachlässigender Teil der Datenanbieter sehr lange Antwortzeiten erreicht [CrBe97, Scha05, SAD+06, VKM03].

In den Experimenten werden die beiden Algorithmen zum Weiterleiten von Anfragen an die Datenanbieter – mit entfernten Nachbarschaftsanfragen (*ENA*) und mit entfernten Gebietsanfragen (*EGA*), siehe Kapitel 4.3.3.5 – mit den vier Methoden zum Schätzen des Radius des initialen Anfragegebiets kombiniert – Null (*Null*), Objektdichte (*ObjDichte*), Objektanzahl (*ObjAnzahl*) und Maximum (*Max*), siehe Kapitel 4.3.3.3. So entstehen acht Varianten, die in zwei Gruppen aufgeteilt werden: die *ENA*-basierten und die *EGA*-basierten Varianten. Das Verhalten der vier Schätzmethoden wird getrennt für jede Gruppe analysiert, und die beste Methode jeder Gruppe wird miteinander verglichen. *EGA_ObjDichte* und *EGA_ObjAnzahl* stellen die Adaption der in [LLN02] vorgestellten Verfahren auf das hiesige Szenario dar. *ENA_ObjAnzahl* simuliert die in [PaMa96] vorgestellten Algorithmen. Die anderen in Kapitel 4.3.2.4 vorgestellten Ansätze werden durch die Varianten *ENA_Max* und *EGA_Max* repräsentiert.

Die Leistung des FNA-Algorithmus wird mit Hilfe von vier Metriken bewertet: Antwortzeit, Aufwand, Anzahl der Iterationen und Kosten-Nutzen-Verhältnis. Die Antwortzeit ist die Summe der Antwortzeiten der befragten Datenanbieter in dem am längsten laufenden Thread in der BEFRAGEANBIETER Funktion. Es wird angenommen, dass der Verarbeitungsaufwand in der Föderationskomponente so klein ausfällt, dass er im Vergleich zu den Antwortzeiten der Datenanbieter vernachlässigt werden kann. In die Antwortzeiten der Datenanbieter fliessen Netzwerklatenzen und Datenbankzugriffe ein, wohingegen die Verarbeitung der Föderationskomponente im Hauptspeicher stattfindet, was wesentlich schneller ist. Die Aufwands-Metrik quantifiziert den Ressourcenverbrauch und berechnet sich aus der gewichteten Summe der Anzahl der befragten Datenanbieter (*befragteAnbieter*) und der Anzahl der übertragenen Objekte (*übertrageneObjekte*), die mit dem durchschnittlichen Pro-Anfrage-Anteil der Antwortzeit (*mittlereKostenProAnfrage*) und dem durchschnittlichen Pro-Ergebnisobjekt-Anteil der Antwortzeit (*mittlereKostenProObjekt*), siehe Tabelle 6 für deren Werte, gewichtet werden.

$$\begin{aligned} Aufwand = {} & (mittlereKostenProAnfrage \cdot befragteAnbieter) + \\ & (mittlereKostenProObjekt \cdot übertrageneObjekte) \end{aligned}$$

Relative Antwortzeit und relativer Aufwand stellen das Verhältnis des zugehörigen absoluten Werts zum minimalen absoluten Wert unter allen betrachteten Varianten für ein bestimmtes k dar. Die Anzahl der Iterationen der Hauptschleife des FNA-Algorithmus entspricht auch der Anzahl der an den Verzeichnisdienst gestellten Anfragen. Das Kosten-Nutzen-Verhältnis ist das Produkt der relativen Antwortzeit und des relativen Aufwands:

$$Kosten = relativerAufwand$$

$$Nutzen = \frac{1}{relativeAntwortzeit}$$

(kürzere Antwortzeit bedeutet höherer Nutzen)

$$\frac{Kosten}{Nutzen} = relativerAufwand \cdot relativeAntwortzeit$$

(niedriger ist besser)

4.3.4.1 Experiment 1: Anzahl der gesuchten Objekte variieren

Im ersten Experiment reicht der Anfrageparameter k von 1 bis 1024. Damit wird untersucht, wie sich der FNA-Algorithmus bei unterschiedlichen Anfragegrößen verhält. In der BEFRAGEANBIETER Funktion wird die *1+log* Einstellung verwendet. Dies ist das Ergebnis der Untersuchungen im nächsten Teilkapitel. Gegenüber den bisherigen Ansätzen aus der Literatur stellt dies schon eine Optimierung dar, da diese alle relevanten Datenanbieter gleichzeitig befragen. Die Ergebnisse dieses Experiments sind in den Abbildungen 14 und 15 zu sehen.

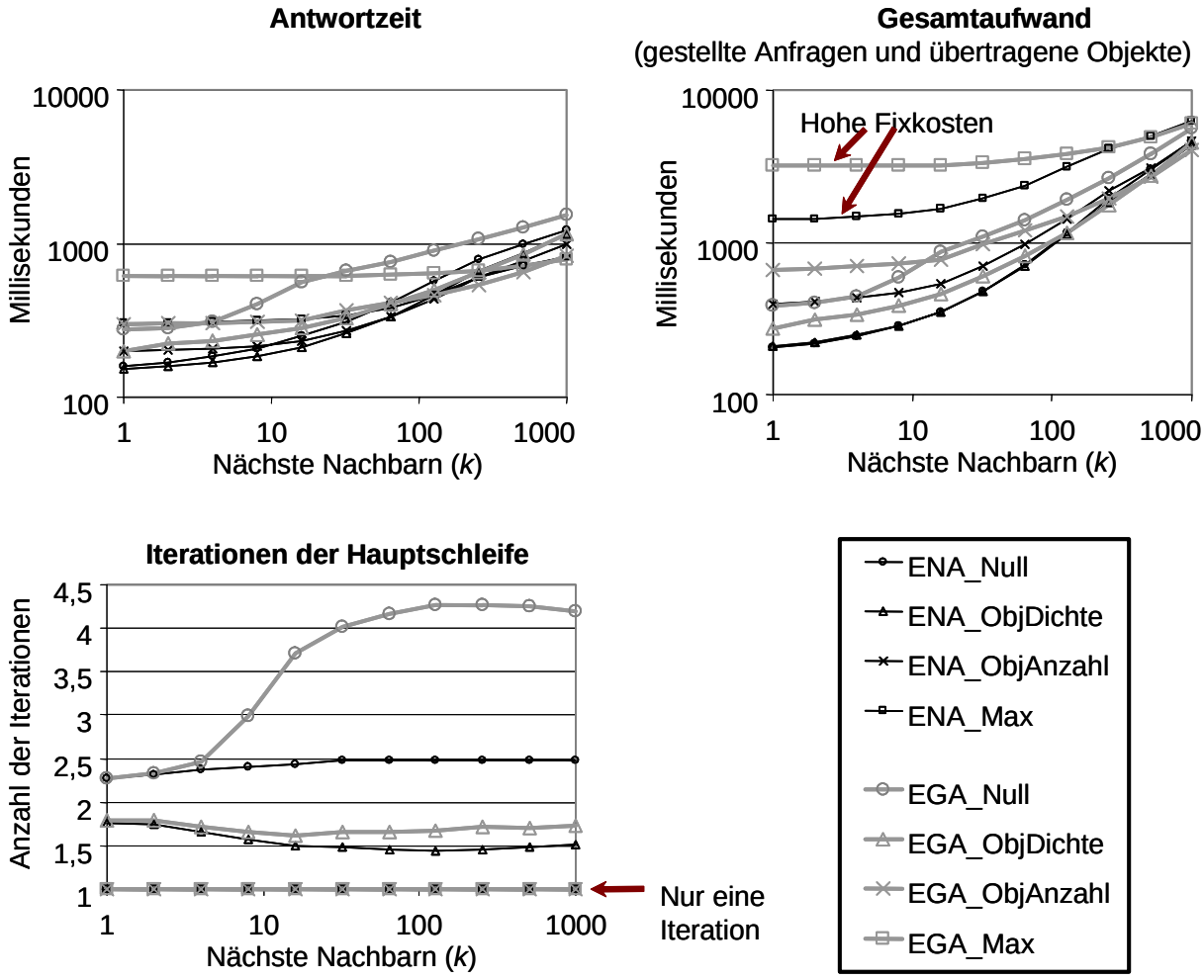


Abbildung 14: Skalierbarkeit des FNA-Algorithmus bei wachsendem k, die X- und Y-Achsen haben eine logarithmische Skala

In Abbildung 14 nehmen die Antwortzeit und der Aufwand mit zunehmendem k nur in sublinearer Weise zu. Der Algorithmus skaliert also sehr gut für große k. Die Anzahl der Iterationen der Hauptschleife ist weitgehend unabhängig von k. Die Varianten mit den *ObjAnzahl* und *Max* Schätzmethoden finden tatsächlich alle Ergebnisobjekte in der ersten Iteration. Bei großen k hat die *EGA_Null* Variante Probleme damit, genügend Objekte zu finden, um eine verlässliche Schätzung der Objektdichte vornehmen zu können, so dass hier mehr Iterationen benötigt werden.

In Abbildung 15 wird deutlich, dass die *ObjDichte* Varianten (sowohl mit entfernen Nachbarschaftsanfragen als auch mit entfernten Gebietsanfragen) die beste Wahl

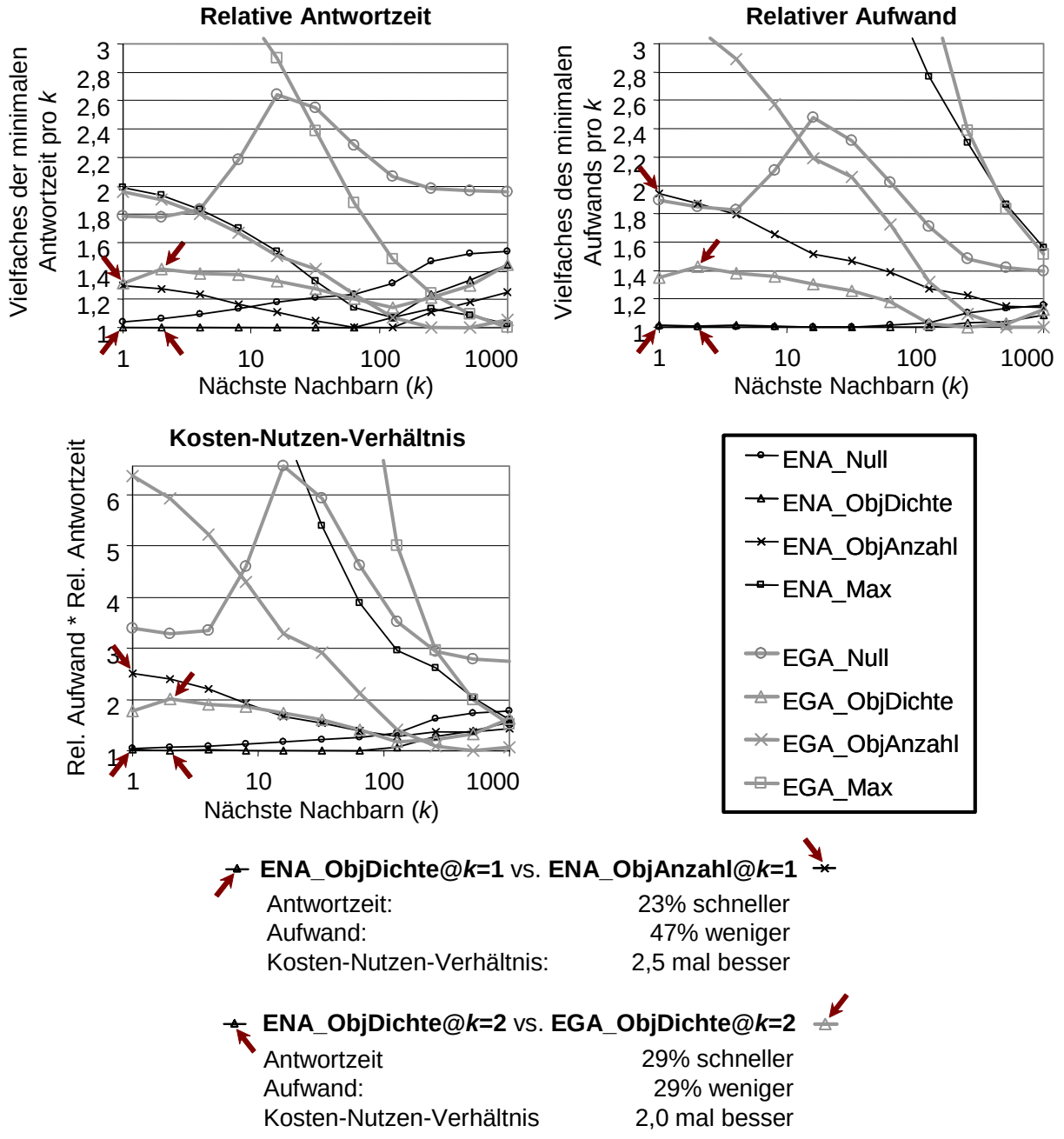


Abbildung 15: Rangordnung der Varianten bei wachsendem *k*. Die Threadanzahl ist auf *1+log* gesetzt. Die Pfeile weisen auf besonders interessante Stellen hin.

darstellen für $k \leq 64$. Die Varianten sind bis zu 23% schneller und haben einen bis zu 47% geringeren Aufwand als die Varianten mit der *ObjAnzahl* Schätzmethode. Die entsprechenden Stellen sind in Abbildung 15 durch Pfeile markiert. Die *ENA_Null* Variante erzielt das zweitbeste Ergebnis für kleine *k*. Bei größeren *k* verschlechtert sich die Leistung auf Grund des folgenden Effekts: Anfangs werden nur wenige Objekte gefunden, so dass wegen der davon abgeleiteten Schätzung der Objektdichte das Anfragegebiet erheblich vergrößert wird, was dazu führt, dass sehr viele Datenanbieter in der folgenden Iteration als relevant erachtet werden und Anfragen an diese gesandt werden. Wenn entfernte Gebietsanfragen zum Einsatz kommen, schneidet die *Null* Schätzmethode sehr viel schlechter ab. Diese ist dann nur bei sehr kleinen *k* die zweitbeste Schätzmethode. Ab $k \geq 8$ werden wesentlich mehr Ite-

rationen benötigt und Antwortzeit und Aufwand verschlechtern sich im Vergleich mit den anderen Varianten. Bei entfernten Nachbarschaftsanfragen schneidet die *Null* Schätzmethode deshalb so gut ab, weil dort nicht die Geometrien der Objekte das Anfragegebiet schneiden müssen, sondern nur das Dienstgebiet des Datenanbieters.

Die Varianten mit den *ObjAnzahl* und *Max* Schätzmethoden haben hohe Fixkosten, die unabhängig von k sind. Dies ist auch im Aufwandsdiagramm in Abbildung 14 zu sehen. Bei kleinen Werten für k hat daher die *Max* Methode einen um Größenordnungen höheren Aufwand als die anderen Methoden. Die *ObjAnzahl* und *Max* Methoden stufen viele Datenanbieter als relevant ein. Damit wird sichergestellt, dass nach nur einer Iteration alle Ergebnisobjekte gefunden sind. Dies hat jedoch den Nachteil, dass sehr viele Anfragen an die Datenanbieter weitergeleitet werden (= hoher Aufwand), und dass die Wahrscheinlichkeit steigt, dass sich unter den relevanten Datenanbietern ein sehr langsamer befindet (= lange Gesamtantwortzeit, da auf dessen Antwort auch gewartet werden muss). Erst bei großen Werten für k ($k \geq 100$) haben die anderen Methoden ähnlich hohe Kosten, so dass der Aufwand und die Antwortzeit der *ObjAnzahl* Methode ein konkurrenzfähiges Niveau erreicht. Bei der *Max* Methode wird die Konkurrenzfähigkeit erst bei noch größeren k erreicht ($k \geq 500$).

Solange $k \leq 128$ ist, sind die Varianten mit entfernten Nachbarschaftsanfragen deutlich schneller (bis zu 29%) und verbrauchen weniger Ressourcen (bis zu 29%) als die Varianten mit entfernten Gebietsanfragen. Die entsprechenden Stellen sind in Abbildung 15 ebenfalls durch Pfeile markiert. Nur bei großen Werten für k ($k \geq 200$) können die *EGA* Varianten die *ENA* Varianten überholen, da erstere von der gleichmäßigen Verteilung der Objekte profitieren können und weniger potentielle Ergebnisobjekte, die letztendlich verworfen werden, von den Datenanbietern übertragen.

4.3.4.2 Experiment 2: Grad der Parallelität variieren

Im zweiten Experiment wird die Anzahl der in der BEFRAGEANBIETER Funktion verwendeten Threads variiert, um den Einfluss der Parallelisierung bei der Weiterleitung der Anfragen an die Datenanbieter auf die Leistung des FNA-Algorithmus zu zeigen. Die Anzahl der Threads ist dabei entweder von vorn herein festgelegt auf einen Wert zwischen 1 und 32, oder hängt dynamisch von der Größe der Liste der zu fragenden Anbieter ab. Im zweiten Fall ist die Threadanzahl entweder ein bestimmter Prozentsatz der Listengröße (*25%*, *33%*, *50%*, *100%*), oder sie ist proportional zum Logarithmus davon (*2*log*, *1+log*).

$$2*\log = 2 \cdot \log_2(\text{AnbieterSL.size})$$

$$1+\log = 1 + \log_2(\text{AnbieterSL.size})$$

Die logarithmischen Einstellungen haben den Vorteil, dass sie bei wenigen zu fragenden Anbieter zu hoher Parallelität führen, so dass nahezu alle der wenigen Anbieter gleichzeitig befragt werden. Bei einer langen Anbieterliste führt der Logarithmus zu einer im Verhältnis zur Länge der Liste moderaten Parallelität, so dass die Abkürzungsmechanismen besser greifen können, ohne ganz auf Parallelität verzichten zu müssen.

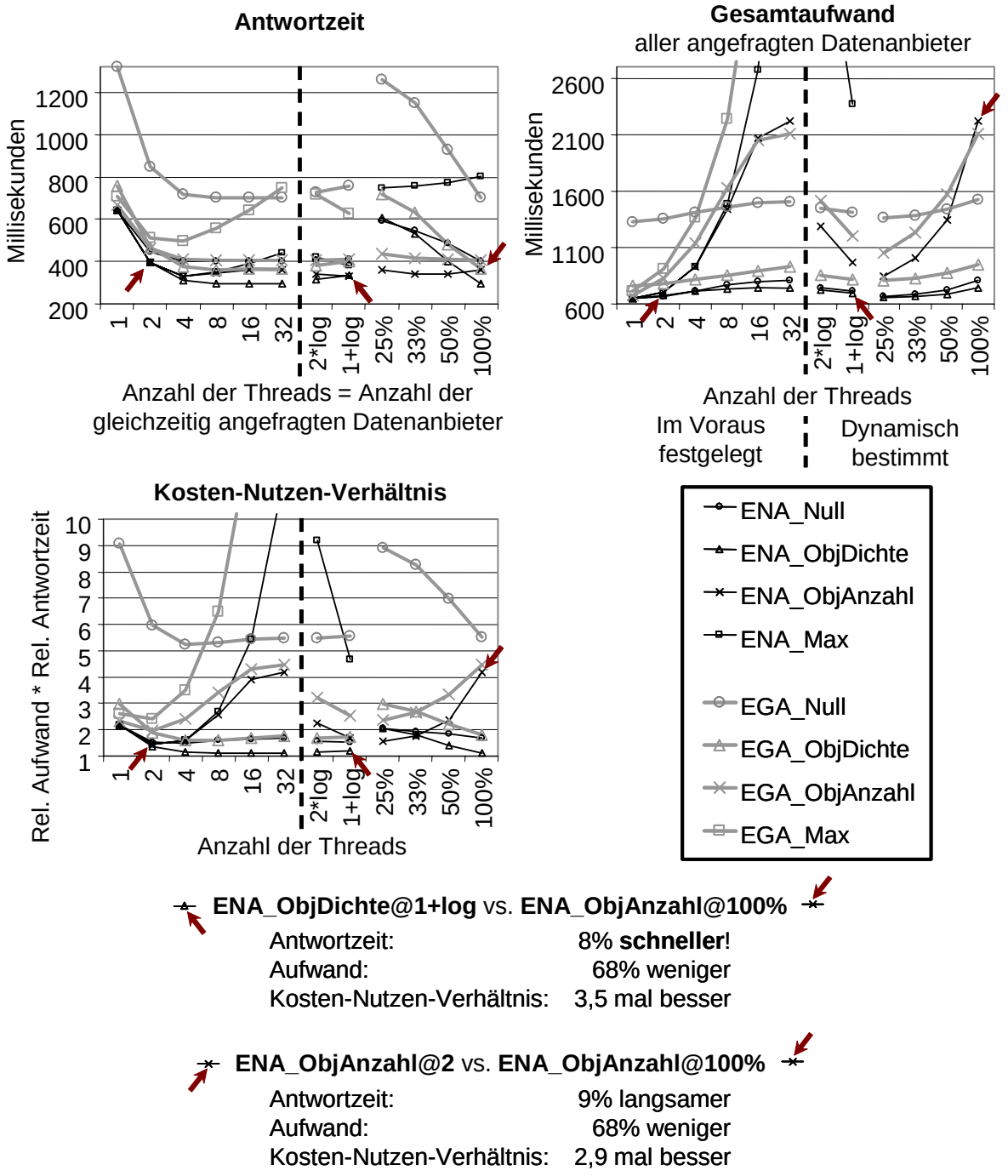


Abbildung 16: Leistung des FNA-Algorithmus wenn die Anzahl der Threads variiert wird. Es werden stets $k = 64$ Objekte gesucht. Die Pfeile weisen auf besonders interessante Stellen hin.

In diesem Experiment wird k fest auf den Wert 64 gesetzt. Der vielversprechendste Ansatz der verwandten Arbeiten ist vergleichbar mit der *ENA_ObjAnzahl* Variante, die alle Datenanbieter gleichzeitig befragt (*100%*). Diese Variante dient als Referenz für die weitere Diskussion. Die Ergebnisse dieses Experiments sind in Abbildung 16 dargestellt.

Offensichtlich führt die Benutzung eines einzigen Threads zu minimalem Aufwand aber langen Antwortzeiten, während die gleichzeitige Befragung aller relevanten Datenanbieter (*100%*) meist zu minimalen Antwortzeiten, jedoch gleichzeitig auch zum größtmöglichen Aufwand führt. Der beste Kompromiss kann erzielt werden,

wenn die Threadanzahl vom Logarithmus der Anzahl der anzufragenden Anbieter abgeleitet wird (*1+log*). Allgemein wird die Antwortzeit kürzer, wenn mehr Threads eingesetzt werden. Nur bei der *Max* Schätzmethode wird die Antwortzeit länger, da hier die Wahrscheinlichkeit steigt, auf die Antwort eines sehr langsamen Datenanbieters warten zu müssen, der bei weniger Threads wegen der Abkürzungsmechanismen gar nicht gefragt worden wäre.

Bei der *ObjDichte* Schätzmethode werden insgesamt nur sehr wenige Datenanbieter als relevant eingestuft, so dass hier die Anzahl der eingesetzten Threads nur eine geringe Auswirkung auf die Leistung hat. Das Gegenteil gilt bei den *ObjAnzahl* und *Null* Schätzmethoden. Erstere stuft wegen der Ein-Iteration-Garantie sehr viele Datenanbieter als relevant ein, die alle auch noch in der gleichen Iteration befragt werden. Bei letzterer wird nach initial viel zu wenigen gefundenen Objekten das Anfragegebiet stark vergrößert, so dass nun sehr viele Datenanbieter relevant sind. In beiden Fällen hat die Threadanzahl einen deutlichen Einfluss auf die Antwortzeit und den Aufwand. Wenn die *Max* Schätzmethode mit vielen parallelen Threads kombiniert wird, so steigt der Aufwand so stark, dass er weit jenseits der in Abbildung 16 dargestellten Skala liegt. Die Paare der Varianten mit jeweils der selben Schätzmethode aber verschiedenen entfernten Anfragetypen zeigen ein ähnliches Verhalten, wenn die Anzahl der Threads variiert wird. Bei $k = 64$ sind die Varianten mit entfernten Nachbarschaftsanfragen immer leicht besser als die entsprechende Variante mit entfernten Gebietsanfragen.

Mit den folgenden Einstellungen lässt sich nach Abbildung 16 ein optimales Kosten-Nutzen-Verhältnis erzielen. Dies wird erreicht, wenn das Produkt aus relativer Antwortzeit und relativem Aufwand 1,0 beträgt. Dann sind die Kosten minimal und der Nutzen maximal:

- **2 Threads**. Führt zu einem sehr niedrigen Aufwand bei einer fast optimalen Antwortzeit. Diese statische Festlegung hängt jedoch ab von der Anzahl der gesuchten Objekte und von der Systemumgebung. Beispielsweise liefert bei $k = 256$ eine Threadanzahl von 4 die besseren Ergebnisse.
- **1+log Threads**. Führt zu einer annähernd optimalen Antwortzeit bei leicht erhöhtem Aufwand. Diese Einstellung liefert auch bei anderen Werten für k vergleichbare Ergebnisse und kann damit unabhängig von k empfohlen werden. Diese Einstellung liefert die besten Ergebnisse von allen dynamischen Threadanzahlvarianten.
- **25%**. Diese Einstellung ist nur zusammen mit der *ObjAnzahl* Schätzmethode empfehlenswert. Sie führt zu einem niedrigen Aufwand bei fast optimaler Antwortzeit und ist dabei unabhängig von k. In Kombination mit den *Null* und *ObjDichte* Schätzmethoden läuft es bei dieser Einstellung darauf hinaus, dass die Datenanbieter meist rein sequentiell befragt werden, weil diese Methoden nur sehr wenige Anbieter als relevant einstufen.
- **100%**. Diese Einstellung ist nur zusammen mit der *ObjDichte* Schätzmethode empfehlenswert, da bei dieser Methode nur sehr wenige Anbieter befragt werden. Führt dabei zu den kürzesten Antwortzeiten bei einem bemerkenswert niedrigen Aufwand. Das Kosten-Nutzen-Verhältnis ist sogar geringfügig besser als bei der *1+log* Einstellung, allerdings nur bei der *ObjDichte* Schätzmethode.

4.3.4.3 Zusammenfassung der Messungen

Die Messungen haben gezeigt, dass die *ObjDichte* Schätzmethode am besten geeignet ist, um den Radius des initialen Anfragegebiets zu ermitteln. Diese Methode übertrifft alle anderen Methoden deutlich solange $k \leq 64$, wobei das Kosten-Nutzen-Verhältnis bis zu 2,5 mal besser ist. Bei größeren Werten für k hat diese Methode immer noch den kleinsten Aufwand. Diese Methode benötigt nur eine Schätzung der globalen Objektdichte, welche relativ einfach erfasst werden kann. Entfernte Nachbarschaftsanfragen (*ENA*) sind entfernten Gebietsanfragen (*EGA*) vorzuziehen, wenn die Datenanbieter eine entsprechende Schnittstelle bereitstellen, da diese besser mit ungleichmäßigen Objektverteilungen fertig werden. Trotzdem ist der Aufwand und die Antwortzeit bei der *EGA_ObjDichte* Variante höchstens 29% schlechter als bei der *ENA_ObjDichte* Variante für $k \leq 128$. Die Einstellung *1+log* für die Threadanzahl erreicht den besten Kompromiss zwischen Antwortzeit, welche höchstens 12% über der minimalen Antwortzeit bei der *100%* Einstellung liegt, und Aufwand, welcher höchstens 79% über dem minimalen Aufwand liegt, der bei einem Thread erreicht wird. Die *1+log* Einstellung liefert bei allen Varianten gleichermaßen gute Leistungen. Wenn alle Stellschrauben kombiniert werden, dann kann die *ENA_ObjDichte* Variante mit der Threadanzahl *1+log* den vielversprechendsten Ansatz aus der Literatur, welcher durch die *ENA_ObjAnzahl* Variante mit der Threadanzahl *100%* repräsentiert wird, um einen Faktor von 3,5 im Kosten-Nutzen-Verhältnis übertreffen.

Wenn man nun in Betracht zieht, dass bei vielen Anwendungen realistische Werte für k in der Größenordnung von 10 liegen, und dass Antwortzeiten im Bereich von unter zwei Sekunden vertretbar sind, dann zeigen die Messungen hier deutlich, dass die Schätzmethoden *Null* und *ObjDichte* die bevorzugten sind. Hierbei bleibt noch genügend Spielraum, um längere Antwortzeiten gegen geringeren Aufwand einzutauschen, was insbesondere bei großen Systemen äußerst wichtig ist.

4.3.5 Zusammenfassung

In diesem Kapitel wurde das Problem der Verarbeitung von Nachbarschaftsanfragen in einer Föderation aus lose gekoppelten Anbietern von räumlichen Daten diskutiert. Es wurde der FNA-Algorithmus vorgestellt, der solche föderierten Nachbarschaftsanfragen (FNA) verarbeitet [SIG+04]. Verschiedene Varianten dieses Algorithmus wurden erläutert und mittels Messungen miteinander verglichen. Insbesondere wurde dabei angestrebt, die Anzahl der anzufragenden Datenanbieter möglichst gering zu halten, was durch dynamisches Vergrößern und Verkleinern des Anfragegebiets, und damit der Menge der relevanten Datenanbieter, erreicht werden konnte. Durch eine Erweiterung der Parametrisierungsmöglichkeiten – Kontrolle des Grads an Parallelität und Bestimmung der relevanten Datenanbieter mittels Dienstgebieten und einer globalen Schätzung der Objektdichte – konnten beträchtliche Leistungssteigerungen gegenüber den bisherigen Ansätzen aus verwandten Arbeiten erzielt werden. Durch die Anpassung der Parallelität kann Antwortzeit gegen Ressourcenverbrauch eingetauscht werden. Die Messungen haben gezeigt, dass der Algorithmus, der entfernte Nachbarschaftsanfragen, einen Schätzwert der globalen Objektdichte sowie *1+log* Threads einsetzt, die beste Variante zur Verarbeitung von föderierten Nachbarschaftsanfragen darstellt. Die Ansätze aus der Literatur können lokal bei einem Datenanbieter eingesetzt werden, um die an

ihn gestellten entfernten Nachbarschaftsanfragen zu verarbeiten. Dennoch kann der FNA-Algorithmus auch ohne entfernte Nachbarschaftsanfragen auskommen, indem diese durch entfernte Gebietsanfragen ersetzt werden. Dies führt zu einem merklichen aber nicht übermäßigen Leistungsverlust. Insgesamt kann gesagt werden, dass diese Diskussion der Verarbeitung von Nachbarschaftsanfragen in einer föderierten Umgebung den bisherigen Wissensstand über Nachbarschaftsanfragen beachtlich voranbringt.

4.4 Behandlung von Mehrfachrepräsentationen

Wenn in einem offenen und verteilten System wie der Nexus-Plattform die Daten vieler verschiedener Datenanbieter föderiert werden, damit die Verteilung der Daten vor den Anwendungen verborgen werden kann, dann ist es sehr wahrscheinlich, dass in den Daten Duplikate enthalten sind. In der Anwendungsdomäne der kontextbezogenen Anwendungen treten Duplikate auf, wenn zwei Datenanbieter Daten über dieselbe Entität in der realen Welt zur Verfügung stellen. Dies wird im Folgenden als Mehrfachrepräsentation oder auch als mehrfach repräsentiertes Objekt bezeichnet.

Die Verarbeitung von Mehrfachrepräsentationen kann in drei Problembereiche gegliedert werden: Zuordnung, Verschmelzung und Anfrageverarbeitung. Bei der Zuordnung werden zusammengehörige Objekte identifiziert, und bei der Verschmelzung werden die verteilten Daten der zusammengehörigen Repräsentationen zu einem Objekt zusammengefasst. Diese beiden Aspekte werden in Kapitel 4.4.1 diskutiert nachdem das allgemeine Konzept der Mehrfachrepräsentationen eingeführt wurde. Bei der Anfrageverarbeitung wird das Problem adressiert, wie die auf eine gegebene Anfrage passenden Objekte ermittelt werden können, wenn deren Daten auf mehrere Anbieter verteilt sind. Hierfür werden in Kapitel 4.4.2 zunächst alle Konstellationen der Datenverteilung theoretisch durchgespielt und deren Auswirkungen auf die Vollständigkeit und Korrektheit der Ergebnismenge untersucht. Es wird ebenfalls untersucht, ob durch eine Abwandlung der Semantik der von der Föderationskomponente an die Datenanbieter weitergeleiteten Anfrage ein Vorteil erzielt werden kann. Die Ergebnisse der Untersuchungen werden in Kapitel 4.4.3 in einen praktischen Algorithmus zur Verarbeitung von Anfragen in der Föderation umgesetzt. In Kapitel 4.4.4 wird der hier vorgestellte Ansatz in Bezug zu verwandten Arbeiten gesetzt. Eine Bewertung der Ergebnisse in Kapitel 4.4.5 rundet diesen Themenbereich ab.

4.4.1 Konzept der Mehrfachrepräsentationen

Eine Mehrfachrepräsentation kann sich auf einzelne Objekte wie ein Hotel oder ein Restaurant beziehen, oder auch auf ganze Gruppen von Objekten wie die Bestandteile eines Straßennetzes, siehe Abbildung 17. Die bei verschiedenen Datenanbietern zu einem Objekt gespeicherten Daten können sich hierbei ergänzen, sich überlappen, oder auch sich in Teilen widersprechen. In allen Fällen soll die Föderationskomponente den Anwendungen eine integrierte und konsistente Sicht auf die Daten bereitstellen, ohne dass die Anwendung von den dabei auftretenden Schwierigkeiten etwas wahrnimmt. Im Folgenden werden die zu einer Entität der realen

Welt bei einem Datenanbieter gespeicherten Daten kurz als Repräsentation bezeichnet. Die Verschmelzung aller zu dieser Entität zugehörigen Repräsentationen wird Objekt genannt.

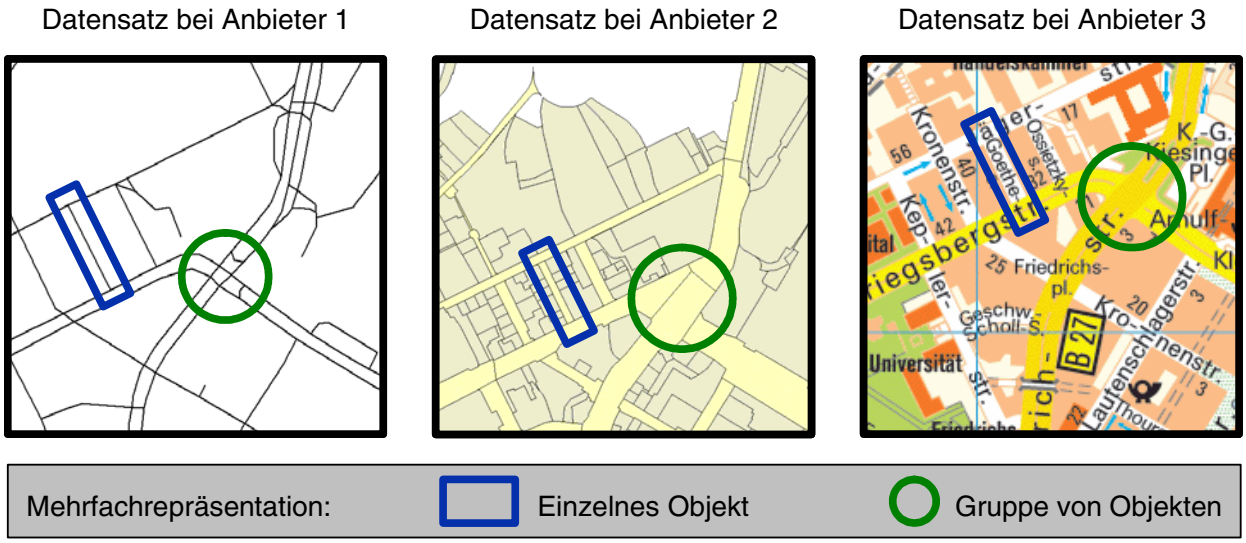


Abbildung 17: Beispiele für Mehrfachrepräsentationen in den überlappenden Datensätzen dreier Anbieter

Mehrfachrepräsentationen lassen sich in zwei Klassen einteilen, die sich hinsichtlich ihrer Eigenschaften und deren Behandlung bei der Anfrageverarbeitung unterscheiden: Zum einen gibt es einzelne Mehrfachrepräsentationen, für welche generische Verfahren in der Anfrageverarbeitung eingesetzt werden können. Zum anderen gibt es Gruppen von Mehrfachrepräsentationen, welche nur mittels domänenspezifischer Verfahren in der Anfrageverarbeitung zusammengeführt werden können.

Bei einzelnen Mehrfachrepräsentationen beziehen sich die bei verschiedenen Datenanbietern gespeicherten Repräsentationen auf ein und dasselbe eindeutige Objekt. Im Bereich der Datenintegration wurde diesem Problem schon einige Aufmerksamkeit gewidmet, und es wurden sowohl generische als auch spezifische Lösungen zur Zuordnung und Verschmelzung von Duplikaten bzw. zur Lösung von Konflikten vorgeschlagen, siehe u. a. [BlNa05]. Die besonderen Probleme, die in der föderierten Anfrageverarbeitung auftreten, wenn die Daten bei ihren Quellen verbleiben sollen, wurden bisher nicht thematisiert. Am verwandtesten zum hier diskutierten Problem ist das Garlic Projekt [Fagi96], wo die Daten zu einem Objekt auf verschiedene Systeme verteilt sind. Jedoch liegt dort der Fokus auf der Verarbeitung von Ähnlichkeitsanfragen, welche die besten k Ergebnisse zurückliefern sollen. Eine weitergehende Diskussion der verwandten Arbeiten folgt in Kapitel 4.4.4.

Im Bereich der kontextbezogenen Anwendungen können für einzelne Mehrfachrepräsentationen die folgenden grundsätzlichen Konstellationen auftreten, siehe Abbildung 18:

- Die verschiedenen Repräsentationen eines Objekts bestehen aus denselben Daten. Hierbei speichern alle Datenanbieter dieselben Attribute des Objekts und zu den Attributen jeweils dieselben Werte.

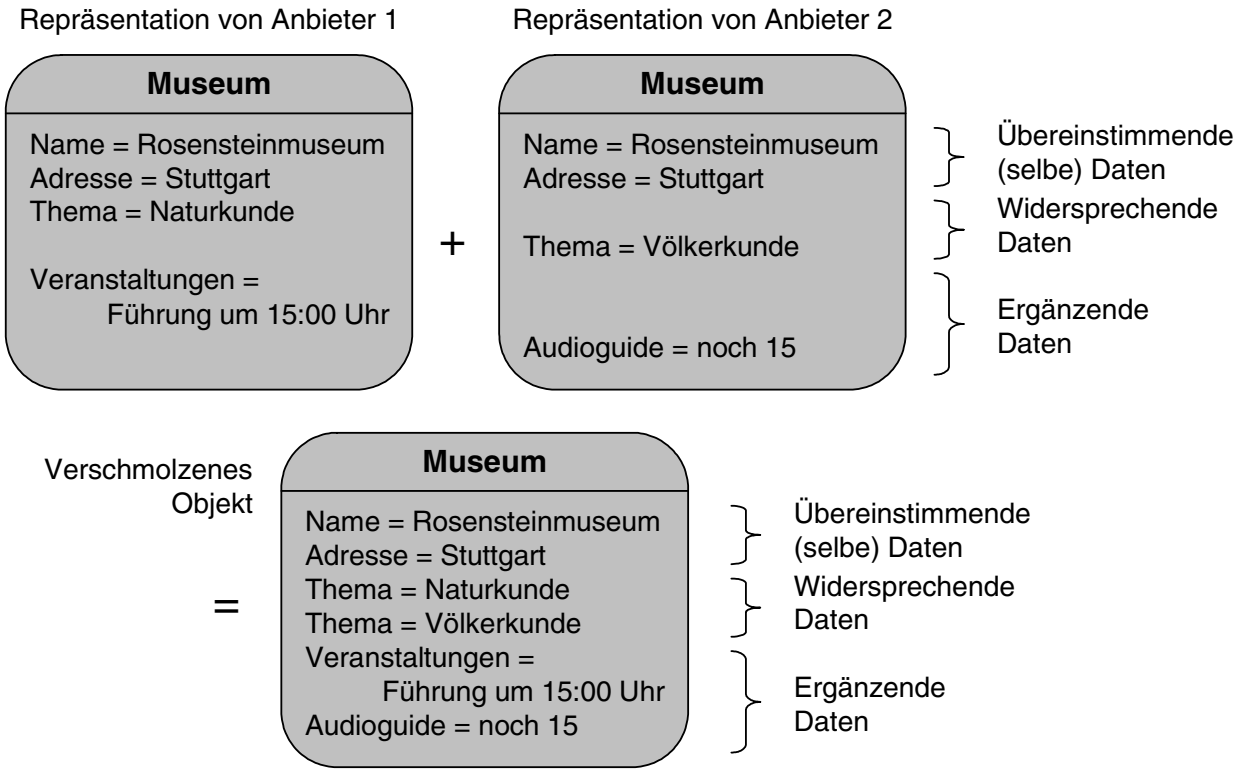


Abbildung 18: Grundsätzliche Konstellationen bei der Verschmelzung zusammengehöriger Repräsentationen

- Die Repräsentationen bestehen aus sich ergänzenden Daten. Hierbei speichern die Datenanbieter verschiedene Attribute des Objekts. Beispielsweise kann ein Datenanbieter das Thema der aktuellen Ausstellung eines Museums kennen und ein anderer die aktuellen Eintrittspreise oder die Verfügbarkeit von sogenannten Audioguides. Bei einem Restaurant könnte ein Datenanbieter dessen Speisekarte und Art der Speisen bereitstellen, während ein anderer Datenanbieter eine Bewertung des Restaurants vorhält.
- Die Repräsentationen enthalten sich widersprechende Daten. Die Datenanbieter stellen dabei verschiedene Werte für dasselbe Attribut des Objekts bereit. Beispielsweise kann ein Datenanbieter einem Straßenabschnitt den Namen „B14“ zuordnen, und ein anderer den Namen „Hauptstätter Straße“. Obwohl die Werte widersprüchlich sind, sind beide richtig. Sie repräsentieren verschiedene Sichtweisen auf das Objekt. Je nach der Situation, in der sich ein Nutzer befindet („Will schnell durch die Stadt“ oder „Sucht ein bestimmtes Haus“) ist der eine oder der andere Name hilfreicher. Da das Datenmodell Mehrfachattribute unterstützt, kann die Föderationskomponente hier beide Werte an die Anwendung weiterleiten und diese den passenderen Wert auswählen lassen.

In der zweiten Klasse von Mehrfachrepräsentationen wird eine Gegebenheit in der realen Welt durch eine Gruppe von Objekten repräsentiert. Die Datenanbieter wählen jedoch eine unterschiedliche Aufteilung der Gegebenheit auf einzelne Objekte, weshalb keine eindeutige 1:1 Zuordnung zwischen einzelnen Repräsentationen möglich ist, sondern nur n:m Zuordnungen zwischen Gruppen von Repräsentationen möglich sind. Straßen, Kreuzungen oder Häusergruppen sind typische Vertreter solcher Gruppen. Drei Beispiele von n:m Zuordnungen sind in Abbildung 19

dargestellt. Bei einer Straße kann jede Fahrbahn als eigenes Objekt oder als nur ein Straßenobjekt mit einer höheren Fahrbahnanzahl dargestellt werden. Verschiedene Datenanbieter können auch unterschiedlich viele einzelne Straßensegmente zur Darstellung des Straßenverlaufs einsetzen. Eine Kreuzung kann unter anderem als einzelner Knoten dargestellt werden, oder es kann jede Abbiegemöglichkeit als eigenes Straßenstück in das Modell aufgenommen werden. Häuser können einzeln oder als zusammengefasster Häuserblock repräsentiert werden.

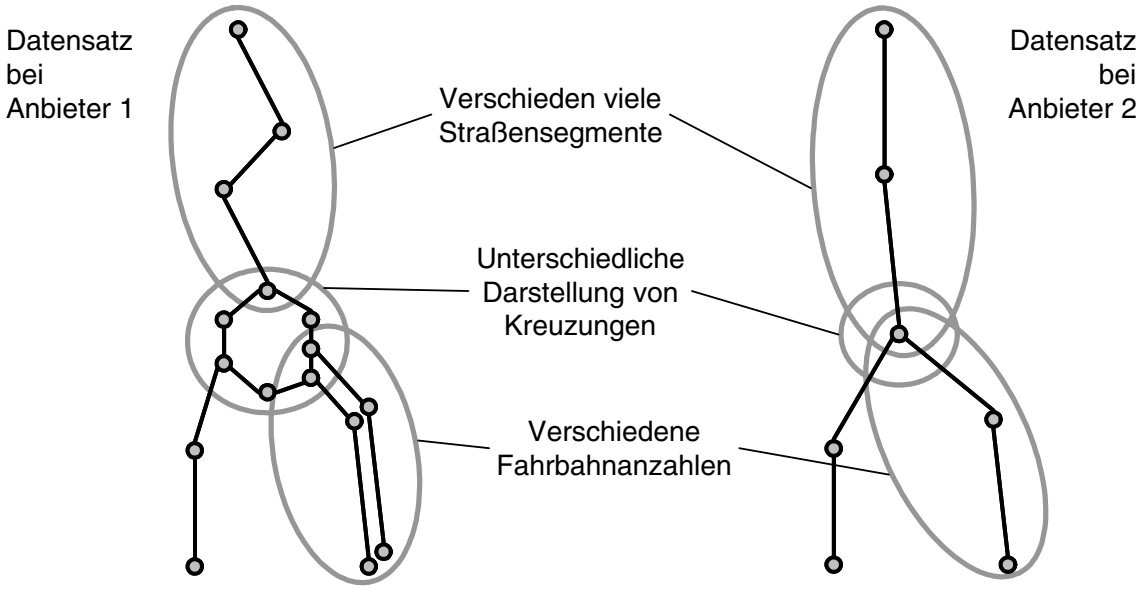


Abbildung 19: Vergleich der Modellierung einer Straße in den Datensätzen zweier Datenanbieter als Beispiel für Mehrfachrepräsentationen, die sich auf Gruppen von Objekten beziehen

Es ist offensichtlich, dass hier nicht mehr von Duplikaten im klassischen Sinne gesprochen werden kann, sondern dass der Duplikatsbegriff auf Gruppen von Objekten ausgedehnt werden muss. Die Zuordnung und Verschmelzung solcher Gruppen von Objekten kann jedoch nicht in generischer Weise erfolgen, sondern erfordert ein tiefgehendes Verständnis der Semantik der Daten. Dies führt zur Entwicklung von vielen spezifischen Verfahren, die jeweils auf einen bestimmten Objekttyp oder ein bestimmtes Einsatzszenario zugeschnitten sind. Diese Problemstellungen sind extrem komplex und füllen eigenständige Dissertationen, z. B. [Walt97], so dass Gruppen von Mehrfachrepräsentationen hier nur sehr oberflächlich behandelt werden können.

4.4.1.1 Zuordnung von Mehrfachrepräsentationen

Bei der Zuordnung von Mehrfachrepräsentationen sollen die Beziehungen zwischen den Repräsentationen einzelner Anbieter herausgefunden werden und somit zusammengehörige Repräsentationen, die sich auf dieselbe Entität in der realen Welt beziehen, identifiziert werden. Die Identifikation der Mehrfachrepräsentationen stellt für sich genommen schon ein schwieriges Problem dar, das deshalb getrennt vom verwandten Problem der Verschmelzung von Mehrfachrepräsentationen diskutiert wird. Im Folgenden werden Ansätze mittels identischer Objekt-IDs, MRep-Relationenobjekten und domänenspezifischen Verfahren diskutiert. Die ers-

ten beiden Ansätze stellen dabei generische Ansätze dar, die unabhängig von der Anwendungsdomäne sind.

Ein Nexus Object Locator (NOL) dient der Identifikation und der Auffindung von Repräsentationen. Ein NOL besteht aus dem Namen des speichernden Servers, einer Identifikationsnummer des logischen Datensatzes, in welchem die Repräsentation enthalten ist (Augmented Area ID, kurz AAID), und einer Identifikationsnummer des Objekts, zu dem die Repräsentation gehört (Objekt-ID). Weitere Details können in Kapitel 2.3 nachgelesen werden. In den folgenden Darstellungen werden symbolische NOLs verwendet, die auf das Wesentliche reduziert sind. Vom Konzept des Datensatzes wird abstrahiert. Wenn ein Datenanbieter verschiedene Datensätze anbietet, so wird im Folgenden jeder Datensatz wie ein eigenständiger Datenanbieter angesehen.

Der einfachste Zuordnungsansatz verwendet identische Objekt-IDs, um zusammengehörige Repräsentationen zu markieren, siehe Abbildung 20. Hierbei müssen die Datenanbieter mithelfen und sich gegenseitig absprechen, indem sie beim Anlegen einer neuen Repräsentation zunächst nach zugehörigen Repräsentationen bei anderen Anbietern suchen und gegebenenfalls deren Objekt-ID in den NOL der neuen Repräsentation übernehmen. Der NOL der neuen Repräsentation wird dadurch eindeutig, dass er zusätzlich zur Objekt-ID auch die eindeutige Identifikationsnummer des Datensatzes und den Namen des speichernden Datenanbieters enthält. Es wird angenommen, dass ein Datenanbieter keine zwei Repräsentationen zur selben Entität in der realen Welt anbieten will, sondern seine Daten vor deren Veröffentlichung konsolidiert. Die Generierungsmethode für neue Objekt-IDs, welche 128 Bit lang sind, schließt durch den Einbezug der Systemzeit und der MAC-Adresse der Netzwerkkarte aus, dass irrtümlicherweise zwei unverbundene Repräsentationen dieselbe Objekt-ID erhalten. Da eine Repräsentation nur einen eindeutigen NOL besitzt, welche den lokalen Primärschlüssel darstellt, eignet sich dieser Ansatz nur für einzelne Mehrfachrepräsentationen.

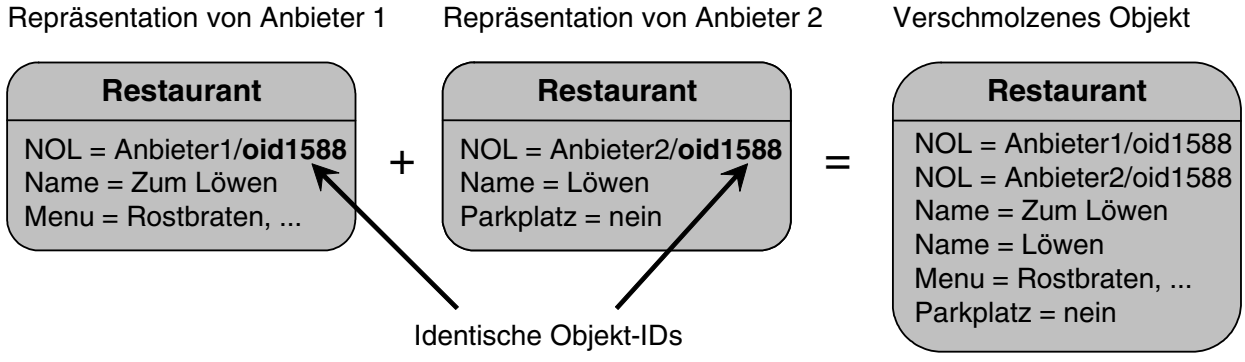


Abbildung 20: Zuordnung mittels identischer Objekt-IDs

Schwergewichtige Relationen, nach der Terminologie von [OGC99b], ermöglichen die Darstellung von Beziehungen zwischen Objekten (dort Features genannt) ohne dass die Objekte selbst hierfür vorbereitet sein müssen und ein entsprechendes Attribut zur Speicherung der Beziehung zur Verfügung stellen müssen. Die Verweise auf die in der Beziehung verknüpften Objekte werden extern in einem separaten Objekt verwaltet. Zur Verknüpfung der Repräsentationen, die zu ein und demselben Objekt gehören, wurden in [ADD+05, VoWa04] sogenannte MRep-Relationenobjekte definiert, welche dem Prinzip einer schwergewichtigen Relation fol-

gen, siehe Abbildung 21. Es können jeweils eine Menge von Quellrepräsentationen mit einer Menge von Zielrepräsentationen verknüpft werden, so dass auf diese Weise die Daten zweier Anbieter in Beziehung gesetzt werden können. MRep-Relationenobjekte können sowohl für einzelne Mehrfachrepräsentationen als auch für Gruppen von Mehrfachrepräsentationen eingesetzt werden. Im Falle einer 1:1 Zuordnung enthält die Relation jeweils nur eine Quell- und eine Zielrepräsentation. Die Zuordnungen zwischen den Repräsentationen von mehr als zwei Datenanbietern müssen auf mehrere MRep-Relationenobjekte aufgeteilt werden, welche jeweils die Daten zweier Anbieter verknüpfen. MRep-Relationenobjekte können selbst eine Geometrie enthalten, damit viele Relationenobjekte auf einmal mit einer räumlichen Anfrage geholt werden können, wie beispielsweise alle Relationenobjekte, die zu den Objekten innerhalb des aktuellen Kartenausschnitts gehören. Zusätzlich können die Repräsentationen selbst Verweise auf die zugehörigen Relationenobjekte enthalten, um einen effizienten Zugriff zu ermöglichen. Diese Verweise werden Rückwärtsverweise genannt. Ein MRep-Relationenobjekt kann sowohl von einem Datenanbieter einer zugehörigen Repräsentation als auch von einem unabhängigen Drittanbieter gespeichert werden.

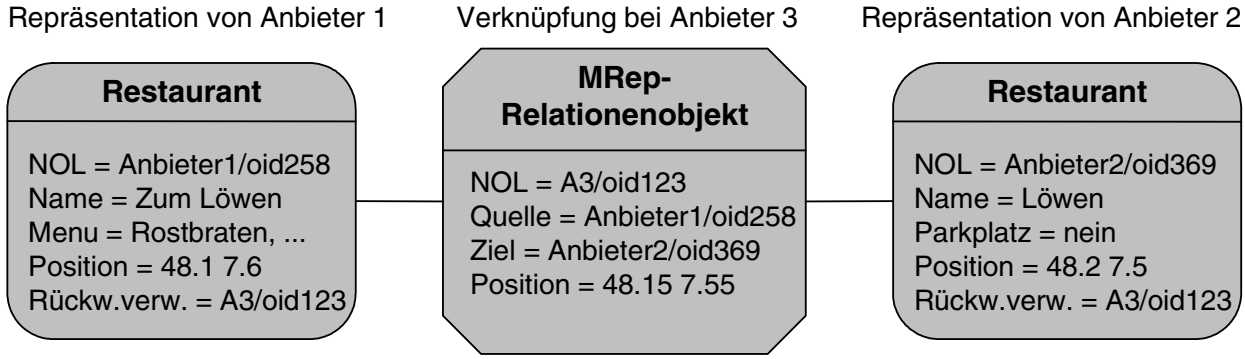


Abbildung 21: Zuordnung mittels MRep-Relationenobjekten (nach [ADD+05, VoWa04])

Als dritte Möglichkeit zur Zuordnung von Mehrfachrepräsentationen stehen domänenspezifische Verfahren zur Verfügung [Walt97, GöSe04, XiSp04, Volz06a]. Diese berechnen die Zuordnungen zwischen den Repräsentationen bei jeder Anfrage neu, und erfordern somit keine explizite Speicherung der Zuordnungsinformationen. Die Verfahren sind speziell zugeschnitten auf die Charakteristika der Anwendungsdomäne. Sie können über einfache Gleichheitsvergleiche hinaus auch die Semantik der Attributwerte in Vergleichen berücksichtigen und somit auch die Konstellation der Repräsentationen zueinander für die Berechnung der Zuordnungen auswerten. Es können domänenspezifische Ähnlichkeitsmaße durch die Kombination der Vergleiche mehrerer Attribute berechnet werden. Ebenso können spezielle Transformationen durchgeführt werden, die z. B. die mittlere Geometrie zweier Repräsentationen berechnen. Domänenspezifische Verfahren können zur Zuordnung sowohl von einzelnen Mehrfachrepräsentationen als auch von Gruppen von Mehrfachrepräsentationen eingesetzt werden. Ein Beispiel für einen domänenspezifischen Zuordnungsalgorithmus (aus [Volz06a]) ist in Abbildung 22 dargestellt. Zum Vergleich zweier Straßen aus verschiedenen Datensätzen wird ein Puffer um deren Geometrien gelegt. Wenn sich die Puffer überlappen dann können die entsprechenden Straßenobjekte zugeordnet werden.

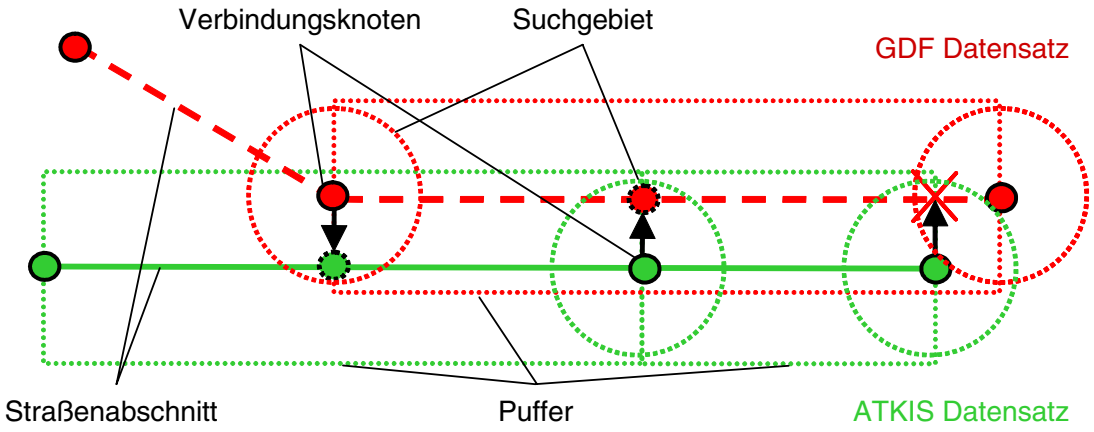


Abbildung 22: Beispiel für einen domänenspezifischen Zuordnungsalgorithmus, aus [Volz06a]

Die ersten beiden Zuordnungsmöglichkeiten setzen voraus, dass die Zuordnungen schon zu einem früheren Zeitpunkt ermittelt worden sind, und in den Objekt-IDs bzw. MRep-Relationenobjekten gespeichert worden sind. Dies kann entweder geschehen, indem eine sachkundige Person die Datensätze mehrerer Anbieter analysiert und vergleicht, oder indem eines der im vorstehenden Absatz beschriebenen domänenspezifischen Verfahren die Zuordnungen automatisch berechnet. In der Anfrageverarbeitung haben identische Objekt-IDs den Vorteil, dass die Zusammengehörigkeit von Repräsentationen schnell überprüft werden kann, wohingegen MRep-Relationenobjekte einen zusätzlichen Anfrageschritt erfordern. Wenn die domänenspezifischen Verfahren zum Anfragezeitpunkt ausgeführt werden führt dies zu einer enormen Rechenlast, da die Berechnungen typischerweise wesentlich komplexer sind, als die gesamte restliche Anfrageverarbeitung, da bei der Suche nach passenden Repräsentationen viele davon paarweise miteinander verglichen werden müssen. Gegenüber den anderen beiden Ansätzen haben MRep-Relationenobjekte den Vorteil, dass alle zu einem Objekt zugehörigen Repräsentationen leicht gefunden werden können, sobald nur eine der Repräsentationen bekannt ist. Ausgehend von dem in der Repräsentation gespeicherten Rückwärtsverweis auf das zugehörige MRep-Relationenobjekt kann dieses direkt abgerufen werden. Anschließend können ausgehend davon alle weiteren Repräsentationen abgerufen werden. Wenn keine Rückwärtsverweise vorhanden sind, können die zugehörigen MRep-Relationenobjekte trotzdem gezielt ermittelt werden, da sie ein Positionsattribut enthalten. Dagegen können bei identischen Objekt-IDs und domänenspezifischen Verfahren nur Zuordnungen zwischen schon in der Föderationskomponente vorliegenden Repräsentationen ermittelt werden. In Kapitel 4.4.2 werden daher nur MRep-Relationenobjekte in der Anfrageverarbeitung berücksichtigt.

4.4.1.2 Verschmelzung von Mehrfachrepräsentationen

Bei der Verschmelzung von Mehrfachrepräsentationen sollen die Daten der einzelnen Repräsentationen zu einem gemeinsamen Objekt zusammengefasst werden, nachdem die Repräsentationen zuvor als zusammengehörig identifiziert wurden. Hierfür stehen zwei Ansätze zur Verfügung: der Mehrfachattributansatz und der domänenspezifische Ansatz.

Der generische und einfachere Ansatz ist der Mehrfachattributansatz. Hierbei werden alle Attributinstanzen der Repräsentationen in das zusammengefasste Objekt übernommen. Falls es Attributinstanzen mit exakt den gleichen Werten gibt, so können die doppelten Attributinstanzen im Objekt ausgelassen werden. Falls es zu einem Attribut mehrere Instanzen mit unterschiedlichen Werten gibt, dann wird dies im zusammengefassten Objekt durch ein Mehrfachattribut dargestellt, siehe auch Abbildung 18 und Abbildung 20. Auf diese Weise können widersprüchliche Werte an die Anwendung weitergeleitet werden, welche dann den passenden Wert selbst bestimmen kann. Dieser Ansatz ist nur bei einzelnen Mehrfachrepräsentationen sinnvoll. Bei Gruppen von Mehrfachrepräsentationen erscheint es wenig gewinnbringend, alle Attributinstanzen der Repräsentationen eines Datenanbieters mit denen eines anderen Datenanbieters zu kombinieren. Der Ansatz ist kombinierbar mit allen drei Ansätzen zur Zuordnung von Mehrfachrepräsentationen. In Kombination mit dem Identische-Objekt-ID- oder MRep-Relationenobjekt-Ansatz kann bei einzelnen Mehrfachrepräsentationen die komplette Verarbeitung auf generische Weise erfolgen. Dies stellt somit eine grundlegende Vorgehensweise dar, die immer funktioniert, auch wenn noch keine domänenspezifischen Funktionalitäten in die Anfrageverarbeitung integriert worden sind.

Alternativ dazu können bei der Verschmelzung von Mehrfachrepräsentationen domänenspezifische Verfahren [Saal87, WaFr99, Haun05, Kund06] eingesetzt werden, um damit eine qualitativ höherwertige Datenintegration zu erreichen. Hierbei können zusätzlich zu einzelnen Mehrfachrepräsentationen auch Gruppen von Mehrfachrepräsentationen verschmolzen werden. Es kann die Konstellation von Repräsentationen zueinander in den Verschmelzungsprozess mit einbezogen werden. Es können je nach Semantik der Objekte und Attribute die passenden Verschmelzungsfunktionen angewandt werden, wie z. B. einen Mittelwert berechen, den Median auswählen, einen Wert auf Grund der Werte anderer Attribute auswählen, oder den längeren Namen wählen. Es können auch komplexe Berechnungen durchgeführt werden, wie z. B. das Berechnen einer mittleren Geometrie [Volz06b], siehe Abbildung 23, oder das gummiartige Verschieben von räumlichen Bereichen (engl. rubber sheeting) [Haun05]. Auch dieser Ansatz ist mit allen drei Ansätzen zur Zuordnung von Mehrfachrepräsentationen kombinierbar.

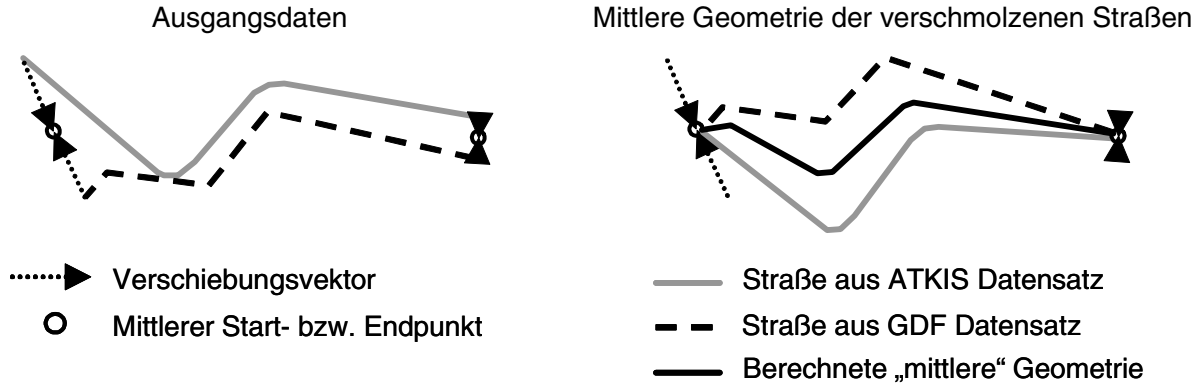


Abbildung 23: Berechnung der mittleren Geometrie als Beispiel für einen domänenspezifischen Verschmelzungsalgorithmus, aus [Volz06b]

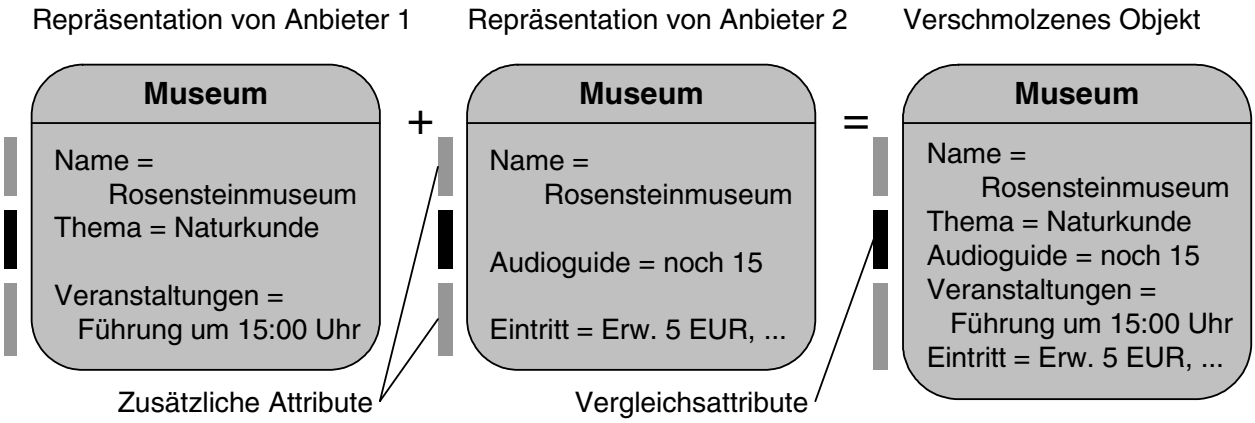
In der Regel ist das Verschmelzen von Mehrfachrepräsentationen wesentlich einfacher als deren Zuordnung, da hier nicht nach passenden Partnern gesucht werden

muss, sondern nur eine vorher bekannte und begrenzte Menge von Repräsentationen zusammengefasst werden muss. Deshalb ist es auch vertretbar, die Repräsentationen bei jeder Anfrage aufs Neue zu verschmelzen und nur deren Zuordnung vorab zu berechnen und in Objekt-IDs oder MRep-Relationenobjekten zu speichern. Die Speicherung der verschmolzenen Objekte hätte die Vorteile, dass die einzelnen Repräsentationen nicht mehr von den Datenanbietern geholt werden müssen, dass der Aufwand zur Durchführung der Verschmelzung wegfällt, und dass die Anfrageverarbeitung leichter die auf eine Anfrage passenden Objekte herausfiltern kann. Dem stehen die Nachteile gegenüber, dass zusätzlicher Speicherplatz benötigt wird, dass bei jeder Änderung der Ausgangsdaten die verschmolzenen Objekte angepasst werden müssen, und dass die Datenanbieter die volle Kontrolle über ihre Daten aufgeben müssen. In jedem Fall stellt das Speichern der verschmolzenen Objekte eine Optimierung dar, die jedoch nicht generell immer angewandt werden sollte, sondern nur an bestimmten Stellen, z. B. bei einem Cache innerhalb einer Föderationskomponente, wenn der Nutzen (schnellere Anfragen) die Kosten (höherer Wartungsaufwand) aufwiegt.

4.4.2 Theoretische Analyse der Anfrageverarbeitung bei Mehrfachrepräsentationen

Die Anfrageverarbeitung in der Föderationskomponente soll alle Objekte finden, die bei globaler Betrachtung auf eine gegebene Anfrage passen. Sie leitet hierzu Anfragen an die relevanten Datenanbieter weiter, um deren Daten zu holen. Solange die Daten zu einem Objekt komplett bei nur einem Datenanbieter gespeichert sind, kann dieser alleine entscheiden, ob das Objekt einen Treffer im Sinne der Anfrage darstellt. Wenn es sich jedoch um ein mehrfach repräsentiertes Objekt handelt, dessen Repräsentationen auf mehrere Datenanbieter verteilt sind, so muss jeder dieser Datenanbieter mittels der ihm vorliegenden Daten entscheiden, ob er die Repräsentation für einen Treffer hält und somit diese als Ergebnis der Anfrage an die Föderation schickt. Im Extremfall, wie in Abbildung 24 dargestellt, sind die Daten so auf die Datenanbieter verteilt, dass kein Datenanbieter seine Repräsentationen als Treffer einstuft, obwohl das Objekt nach der Zusammenfassung aller seiner Repräsentationen tatsächlich doch ein Treffer wäre.

In diesem Kapitel soll nun untersucht werden, welche Spezialfälle in diesem Problemfeld auftauchen können. Es wird diskutiert, wie die Föderationskomponente diese Spezialfälle behandeln kann, und es wird analysiert, welche Auswirkungen dies auf die Leistung des Gesamtsystems hat. Hierfür werden zunächst in Kapitel 4.4.2.1 die Grenzen der folgenden Diskussion abgesteckt. Anschließend werden die Spezialfälle aufgezeigt, die bei mehrfach repräsentierten Objekten auftreten, wenn die Anfrage aus nur einem einzigen Prädikat besteht (Kapitel 4.4.2.2). Es wird untersucht, wie Anfragen, die eine Konjunktion von Prädikaten enthalten, verarbeitet werden können (Kapitel 4.4.2.3). Bei einer Anfrage, die aus einer Disjunktion von Konjunktionen besteht, können die einzelnen Konjunktionen getrennt voneinander jeweils wie eine einzelne Konjunktion verarbeitet werden. Die Verarbeitung einer Disjunktion ergibt somit keine neuen Probleme und wird daher nicht weiter diskutiert. Anschließend werden die Vereinfachungen erläutert, die sich ergeben, wenn zusammengehörige Repräsentationen keine Widersprüche enthalten (Kapitel 4.4.2.4). Im folgenden Kapitel 4.4.3 werden die Betrachtungen in einem



Anfrage: Hole alle Museen, deren Thema Naturkunde ist und die Audioguides verfügbar haben.

Ergebnis: Erfüllt die Repräsentation die Anfrage?

Nein, (Anbieter hat keine Info über Audioguides)	**Nein**, (Anbieter hat keine Info über das Thema)	**Ja**, (Aber kein Datenanbieter schickt eine Repräsentation an die Föderation)

Abbildung 24: Beispiel eines Objekts, dessen Daten auf verschiedene Anbieter verteilt sind, und das deshalb bei einer Anfrage nicht gefunden wird.

Algorithmus zusammengefasst, mit dem die Föderation Anfragen auf Mehrfachrepräsentationen mit Mehrfachattributen verarbeiten kann.

4.4.2.1 Einschränkungen

In diesem Kapitel wird davon ausgegangen, dass zusammengehörige Repräsentationen mittels MRep-Relationenobjekten zugeordnet werden und dass die Verschmelzung mittels des Mehrfachattributansatzes erfolgt. Mit den anderen in Kapitel 4.4.1.1 beschriebenen Zuordnungsansätzen ist es nicht möglich, ausgehend von einer Repräsentation nach den restlichen zugehörigen Repräsentationen zu suchen. Wenn domänenspezifische Verfahren zur Verschmelzung der Repräsentationen eingesetzt werden, dann erschwert dies auch in erheblichem Umfang die Auswahl der auf eine Anfrage passenden Repräsentationen bei einem Datenanbieter, da die Attributwerte vor und nach der Verschmelzung verschieden sein können. Die von einer Anwendung an die Föderationskomponente gestellte Anfrage muss je nach dem zur Verschmelzung genutzten domänenspezifischen Verfahren in eine entsprechende lokale Anfrage transformiert werden. Da auf die domänenspezifischen Verschmelzungsverfahren hier nicht näher eingegangen wird, werden diese auch in der Anfrageverarbeitung nicht berücksichtigt. Aus dem gleichen Grund wird in diesem Kapitel nur die Anfrageverarbeitung für einzelne Mehrfachrepräsentationen beleuchtet, nicht aber für Gruppen von Mehrfachrepräsentationen, die ebenfalls nur mit domänenspezifischen Verfahren zugeordnet und verschmolzen werden können.

Wenn keine Informationen über die Zusammengehörigkeit von mehrfach repräsentierten Objekten vorliegen, dann hat dies zur Folge, dass die Ergebnismenge zu einer Anfrage unvollständig sein kann sowie Duplikate oder sogar irrtümliche Tref-

fer enthalten kann. Die Ergebnismenge ist unvollständig, wenn Objekte nicht gefunden werden, weil die zugehörigen Repräsentationen alleine die Anfrage nicht erfüllen. Hier fehlt das ganze Objekt. Sie ist auch unvollständig, wenn aus dem selben Grund einzelne Repräsentationen nicht einem Trefferobjekt zugeordnet werden können. Hier fehlen Teile der Informationen zu einem Objekt. Duplikate entstehen, wenn mehrere Repräsentationen die Anfrage erfüllen und deren Zusammengehörigkeit auf Grund der fehlenden Informationen nicht festgestellt werden kann. In diesem Fall tauchen die Repräsentationen als eigenständige Objekte, und somit als Duplikate, in der Ergebnismenge auf. Irrtümliche Treffer treten auf, wenn eine einzelne Repräsentation die Anfrage erfüllt, nicht jedoch das gesamte Objekt, weil es von zusätzlichen Informationen aus anderen Repräsentationen disqualifiziert wird. Wenn die anderen Repräsentationen nicht zugeordnet werden können erscheint die erste Repräsentation als irrtümlicher Treffer in der Ergebnismenge.

Des Weiteren wird von dem tatsächlichen Aufbau der Attribute abstrahiert, die aus mehreren Teilen zusammengesetzt sein können [BDG+04, Nick05], siehe Kapitel 2.3. So kann z. B. das Adress-Attribut aus den Teilen Straße, Postleitzahl und Stadt bestehen. Eine Konjunktion von Prädikaten auf verschiedenen Teilen desselben Attributs kann im Sinne des hier diskutierten Problems als ein (komplexes) Prädikat auf einem einfachen Attribut, das aus nur einem Teil besteht, angesehen werden. Somit muss dieser Fall nicht gesondert berücksichtigt werden.

Zusätzlich wird in diesem Kapitel nur der Selektionsteil einer Anfrage betrachtet. Die weiteren in Kapitel 3.3 vorgestellten Teile einer Anfrage haben keinen Einfluss auf das hier diskutierte Problemfeld und werden daher ausgeblendet. Wenn im Folgenden von „Anfrage“ die Rede ist, so ist der Selektionsteil einer Anfrage gemeint. Der Begriff „Anfrageverarbeitung“ bezieht sich auf die Verarbeitung des Selektionsteils einer Anfrage.

4.4.2.2 Verarbeitung eines einzelnen Prädikats bei Mehrfachrepräsentationen

In diesem Kapitel sollen die Schwierigkeiten ergründet werden, die bei der Verarbeitung von Anfragen in der Föderationskomponente durch die verteilte Speicherung der Objekte entstehen. Dabei wird zunächst davon ausgegangen, dass eine Anfrage nur ein einzelnes Prädikat enthält. Zur Vereinfachung der Darstellung wird außerdem davon ausgegangen, dass die Objekte auf nur zwei Datenanbieter verteilt seien. Eine Verallgemeinerung auf beliebig viele Datenanbieter ist leicht möglich, da keine weiteren Sonderfälle auftreten.

Der Sachverhalt soll mit einer an eine Föderationskomponente gestellten Anfrage mit einem verallgemeinerten Prädikat erläutert werden. Das Prädikat bezieht sich auf das Attribut B und wird zu wahr ausgewertet, wenn ein gegebener Wert mit dem Vergleichswert X übereinstimmt. Ein Objekt kann für dieses Attribut entweder keinen Wert haben (0), den gesuchten Wert haben (X), einen anderen als den gesuchten Wert haben (Y) oder sowohl den gesuchten als auch einen anderen Wert haben (XY). Beispielsweise könnte das Ausstellungsthema eines Museumsobjekts nur „Archäologie“, nur „Naturkunde“ oder „Archäologie“ und „Naturkunde“ sein. Zusätzlich speichert ein Objekt weitere von der Anfrage geforderte Daten im Attribut C. Diese werden nicht zur Auswertung des Prädikats benötigt, sollen aber vollständig (von beiden Datenanbietern) in der an die Anwendung zurückgeschickten Ergebnismenge enthalten sein. Im Beispiel könnte dies die Adresse des Muse-

ums, die Öffnungszeiten oder eine Liste mit Sonderveranstaltungen sein. Bezüglich der Anfrage verwaltet jeder Datenanbieter vier Arten von Objekten (*0*, *X*, *Y*, *XY*). Bei der Verteilung der Objektdaten auf zwei Datenanbieter können also prinzipiell 16 Kombinationen auftreten, wovon einige bis auf die Vertauschung der beiden Datenanbieter jeweils paarweise identisch sind. In den Abbildungen 26 bis 29 werden jeweils alle Kombinationsmöglichkeiten untersucht.

Die Föderationskomponente kann jede der in Kapitel 4.1.2 eingeführten Anfragesemantiken an ihrer Anfrageschnittstelle anbieten. Die von ihr an die Datenanbieter weitergeleiteten Anfragen können ebenfalls von allen Semantiken Gebrauch machen und müssen nicht mit der Semantik der an die Föderation gestellten Anfrage übereinstimmen. Diese Freiheit bietet der Föderationskomponente Spielraum für Optimierungen. Der hier relevante Teil der Anfrageverarbeitung in der Föderationskomponente kann in drei Schritte aufgegliedert werden, siehe Abbildung 25, wobei jeder Schritt eine zusätzliche Interaktion zwischen Föderationskomponente und Datenanbietern bedeutet und somit zusätzlichen Aufwand verursacht:

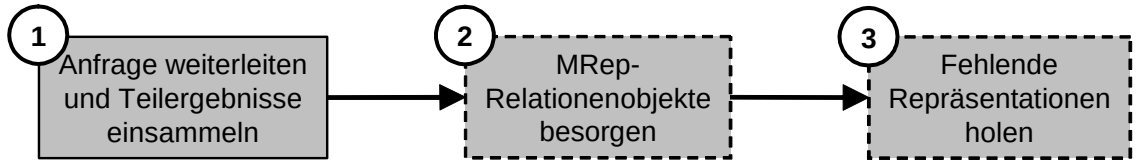


Abbildung 25: Abstrakter Ablauf der Anfrageverarbeitung für Mehrfachrepräsentationen

1. Anfragen an Datenanbieter senden bzw. weiterleiten und deren Ergebnisse einsammeln.
2. Falls notwendig, MRep-Relationenobjekten zu den von den Datenanbietern gelieferten Repräsentationen besorgen. Hierfür können die bei den Repräsentationen gespeicherten sogenannten „reverse references“ (Rückwärtsverweise auf die MRep-Relationenobjekte) genutzt werden. Eine ausführliche Beschreibung dieses Konzepts kann in [ADD+05] nachgelesen werden.
3. Falls notwendig, fehlende Repräsentationen, auf die in den MRep-Relationenobjekten verwiesen wird, von den Datenanbietern holen.

Je nach Verteilung der Daten eines Objekts auf die beiden Datenanbieter können bei der Verarbeitung der Anfrage in der Föderationskomponente die folgenden Fälle eintreten. Am Ende jedes Falls wird der Aufwand abgeschätzt, den die Föderationskomponente bei der Verarbeitung eines solchen Objekts hat.

- **Volltreffer (V)**. Ein Objekt wird als Volltreffer bezeichnet, wenn das Objekt die Anfrage erfüllt und wenn beide Datenanbieter ihre Repräsentation an die Föderation weitergeleitet haben. Dies ist der einzige Fall, bei dem die Zusammengehörigkeit der Mehrfachrepräsentationen direkt in der Föderationskomponente ohne die Hilfe weiterer Daten ermittelt werden kann. Alternativ können hierfür auch die zum Objekt zugehörigen MRep-Relationenobjekte eingesetzt werden. Die Föderationskomponente muss diese jedoch zunächst mit einer weiteren Anfrage beschaffen.

Aufwand: 1 Schritt (Zuordnung der Repräsentationen direkt bestimmen), oder 2 Schritte (Zuordnung mit MRep-Relationenobjekten).

- **Teiltreffer (T)**. Ein Objekt wird als Teiltreffer bezeichnet, wenn das Objekt die Anfrage erfüllt und wenn nur einer der beiden Datenanbieter die entsprechenden Repräsentationen liefert. Um die fehlenden Daten des anderen Datenanbieters holen zu können, muss die Föderationskomponente zunächst die zugehörigen MRep-Relationenobjekte besorgen. Anschließend können die in dem MRep-Relationenobjekten enthaltenen Verweise verfolgt und die fehlenden Repräsentationen geholt werden.

  Aufwand: 3 Schritte.

- **Fehlendes Objekt bzw. Irrtümlicher Ausschluss (F)**. Ein Objekt wird als irrtümlicher Ausschluss (engl. false negative) bezeichnet, wenn das Objekt, insbesondere nach der Verschmelzung all seiner Repräsentationen, die Anfrage erfüllt, aber keiner der Datenanbieter die entsprechende Repräsentation als Treffer einschätzt und an die Föderationskomponente weiterleitet. Da die Föderationskomponente nichts über die Existenz dieses Objekts weiß, kann sie auch nicht die zugehörigen MRep-Relationenobjekte ermitteln. Das Objekt fehlt somit in der an die Anwendung zurückgeschickten Ergebnismenge. Diesen Fall gilt es zu vermeiden!

  Aufwand: keiner, aber fehlerhaftes Ergebnis.

- **Echter Ausschluss (leer)**. Ein Objekt wird als echter Ausschluss bezeichnet, wenn das Objekt die Anfrage nicht erfüllt, und wenn keiner der Datenanbieter die entsprechende Repräsentation an die Föderation liefert. Das Objekt fehlt zu Recht in der Ergebnismenge.

  Aufwand: keiner.

- **Irrtümlicher Treffer (I)**. Ein Objekt wird als irrtümlicher Treffer (engl. false positive) bezeichnet, wenn es als Treffer gewertet wird, obwohl es bei Beachtung aller seiner auf verschiedene Anbieter verteilten Repräsentationen nicht auf die Anfrage passt. Dieser Fall tritt ein, wenn ein Anbieter eine Repräsentation auf Grund der lokal vorhandenen Informationsfragmente als Treffer einstuft und an die Föderation weiterleitet. Ein anderer Anbieter stellt mittels seiner lokalen Informationen fest, dass seine zugehörige Repräsentation nicht auf die Anfrage passt und leitet deshalb diese auch nicht an die Föderation weiter. Der Föderation fehlt nun diese Information zur Disqualifikation des Objekts, so dass das Objekt fälschlicherweise als Treffer gewertet wird.

  Dieser Fall kann zwei Ursachen haben. Zum einen kann er bei der Alle-Anfragesemantik eintreten, bei der zusätzliche Informationen zum Fehlschlagen des Prädikats führen können (wegen des All-Quantors). Zum anderen kann er eintreten, wenn die Föderation eine schwächere bzw. weniger selektive Anfrage an die Datenanbieter schickt als die von der Anwendung an die Föderation geschickte Anfrage. Je nach Verteilung der Informationen können irrtümliche Treffer auf verschiedene Weisen erkannt und beseitigt werden:

  – **Sofort (I-S)**. Die Föderation prüft die von einem Datenanbieter gelieferten Repräsentationen mit der von der Anwendung geforderten Anfragesemantik nach und stellt dabei fest, dass die gelieferte Repräsentation doch kein

Treffer ist, zumindest solange nicht noch weitere Daten von anderen Repräsentationen dazukommen.

Aufwand: 1 Schritt, Repräsentation nutzlos an Föderation übertragen.

- **MRep-Relationenobjekte (I-M)**. Die Föderation besorgt die zu einer Repräsentation zugehörigen MRep-Relationenobjekte und kann damit schon die aktuell betrachtete Repräsentation ausschließen. Hierfür werden zunächst die im MRep-Relationenobjekt genannten und schon in der Föderation vorhandenen Repräsentationen zu einem vorläufigen Objekt verschmolzen. In zwei Situationen kann schon das vorläufige Objekt ausgeschlossen werden. Dies geschieht zum einen, wenn eine Verknüpfung zu einem anderen „sofort“ ausgeschlossenen irrtümlichen Treffer hergestellt werden kann und wenn weitere Daten aus zugehörigen aber noch nicht integrierten Repräsentationen nicht mehr zur Erfüllung der Anfrage beitragen können. Die fehlenden Repräsentationen müssen somit auch nicht beschafft werden. Zum anderen geschieht dies, wenn das vorläufige Objekt schon alle zugehörigen Repräsentationen enthält und trotzdem die Anfrage nicht erfüllt.

  Aufwand: 2 Schritte, Repräsentation und MRep-Relationenobjekte werden nutzlos an Föderation übertragen.

- **Zusätzliche Repräsentationen (I-Z)**. Die Föderation stellt nach dem Beschaffen der MRep-Relationenobjekte zu einer Repräsentation fest, dass noch nicht alle zugehörigen Repräsentationen in der Föderationskomponente verfügbar sind. Nachdem die fehlenden Repräsentationen geholt wurden kann die Föderation alle zusammengehörigen Repräsentationen zu einem Objekt verschmelzen, und in diesem Fall erst danach feststellen, dass dieses Objekt die Anfrage nicht erfüllt.

  Aufwand: 3 Schritte, Repräsentation, MRep-Relationenobjekte und zusätzliche Repräsentationen werden nutzlos an Föderation übertragen.

Die meisten irrtümlichen Treffer können nur entdeckt und beseitigt werden, wenn es die zugehörigen MRep-Relationenobjekte gibt. Ohne diese externen Zuordnungsinformationen können die irrtümlichen Treffer nur im I-S Fall entdeckt werden. In den anderen Fällen enthält die Ergebnismenge unvollständige Objekte, die nur wegen der fehlenden Informationen irrtümlicherweise die Anfrage erfüllen.

Wenn beide Anbieter ihre Repräsentation zu einem Objekt an die Föderation liefern, dann wird nur der aufwändigere Fall dargestellt. Wenn also beispielsweise die Repräsentation von Anbieter 1 sofort ausgeschlossen werden kann und die Repräsentation von Anbieter 2 erst nachdem mittels eines MRep-Relationenobjekts eine Verknüpfung zur Repräsentation von Anbieter 1 hergestellt wurde, dann wird der aufwändigere Fall „I-M“ angezeigt.

Im Folgenden wird systematisch analysiert, wie die Föderationskomponente die Ergebnismenge zu einer an sie gestellten Anfrage durch Anfragen an die relevanten Datenanbieter ermitteln kann. Hierfür wird für jede mögliche Anfragesemantik der eingehenden Anfrage abhängig von der Semantik der an die Datenanbieter geschickten Anfrage untersucht, wie vollständig die Ergebnismenge ist und wie groß der Aufwand ist, diese zu bestimmen. In Abbildung 26 ist eine solche Analyse für eine eingehende Anfrage mit der Existiert-Streng-Semantik dargestellt. Die

Abbildung ist in fünf Teile gegliedert. Teil (a) zeigt das Sollergebnis. Die restlichen vier Teile zeigen die Folgen je einer der vier möglichen Anfragesemantiken (siehe Definition 4.7 bis 4.10 in Kapitel 4.1.2), welche die Föderationskomponente für Anfragen einsetzen kann, die an die Datenanbieter geschickt werden. In jedem Teil sind als Beschriftung der Achsen für jeden der beiden Datenanbieter die möglichen Kategorien von Repräsentationen in Bezug auf die Anfrage dargestellt (*0*, *X*, *Y* und *XY*, wie zuvor eingeführt). Wenn ein Anbieter die Repräsentation im ersten Schritt aus Abbildung 25 an die Föderation liefert, ist die entsprechende Achsenbeschriftung fett gedruckt, sonst kursiv. Die Felder im Kern jedes Teils stellen dar, welcher der zuvor beschriebenen Fälle bei der jeweiligen Verteilung der Daten auf die beiden Anbieter eintritt.

Betrachten wir beispielsweise in Abbildung 26 (b) das Feld am Schnittpunkt der *Y*-Spalte von Anbieter 1 mit der *X*-Zeile von Anbieter 2, welches mit „*T*“ markiert ist. Dieses Feld repräsentiert alle Objekte, bei denen die von Anbieter 1 gespeicherte Repräsentation einen anderen als den gesuchte Wert (*Y*) in Attribut *B* enthält und die von Anbieter 2 gespeicherte Repräsentation nur den gesuchten Wert (*X*) in Attribut *B* enthält. Das verschmolzene Objekt enthält also sowohl den gesuchten Wert als auch einen anderen als den gesuchten Wert (*XY*) in Attribut *B*, und gehört somit bei einer Anfrage mit der Existiert-Streng-Semantik in deren Ergebnismenge. Da nur Anbieter 2 seine Repräsentation an die Föderation schickt (*X* ist fett gedruckt), nicht aber Anbieter 1 (*Y* ist kursiv gedruckt), handelt es sich um einen Teiltreffer (*T*). Damit die Ergebnismenge ein vollständiges verschmolzenes Objekt enthält, das auch die Informationen von Anbieter 1 bezüglich z. B. der Sonderveranstaltungen des Museums enthält, muss die Föderation zunächst mit Hilfe von MRep-Relationenobjekten herausfinden, dass noch weitere Repräsentationen (die von Anbieter 1) zu diesem Objekt gehören. Anschließend müssen die fehlenden Daten von Anbieter 1 geholt werden. Die restlichen Felder decken alle weiteren möglichen Verteilungen der Werte des Attributs *B* eines Objekts ab.

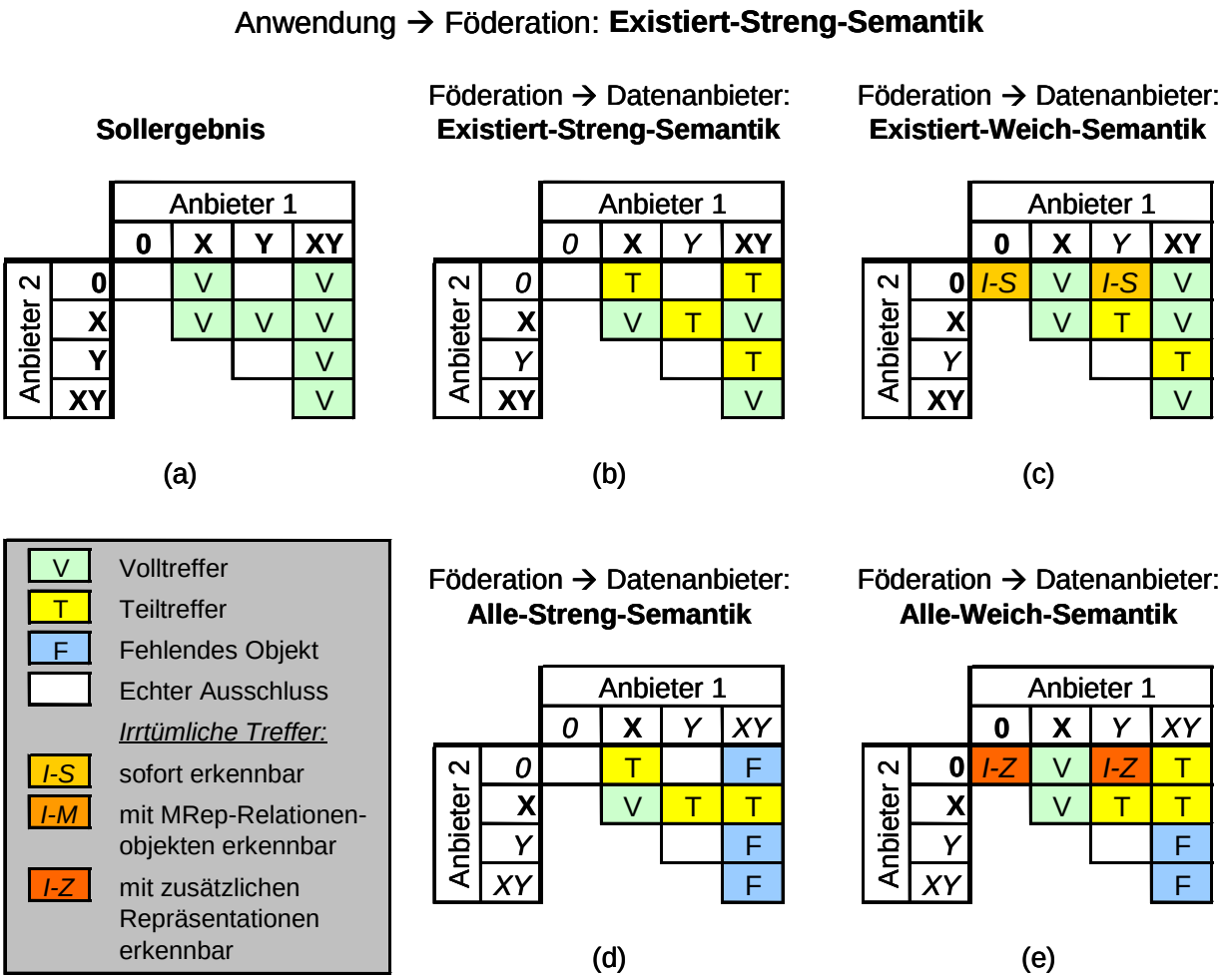


Abbildung 26: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer Anfrage mit Existiert-Streng-Semantik in der Föderation

In Abbildung 26 werden an die Föderation gerichtete Anfragen mit Existiert-Streng-Semantik untersucht. Wenn die Föderation Anfragen mit Existiert-Streng-Semantik an die Datenanbieter schickt (E,S-Anfragen), dann führt dies zu Teiltreffern in einigen Konstellationen. Es gibt jedoch keine irrtümlichen Treffer. Wenn die Föderation Anfragen mit Existiert-Weich-Semantik an die Datenanbieter schickt (E,W-Anfragen), dann gibt es bei mehr Konstellationen Volltreffer, dafür gibt es auch einige irrtümliche Treffer, die jedoch sofort erkannt werden können. Anfragen mit der Alle-Streng- oder Alle-Weich-Semantik (A,S-Anfragen oder A,W-Anfragen) an die Datenanbieter zu schicken ist hier nicht sinnvoll, da beide zu unvollständigen Ergebnismengen mit fehlenden Objekten führen. Somit verarbeitet die Föderation E,S-Anfragen am besten, indem sie E,W-Anfragen an die Datenanbieter schickt.

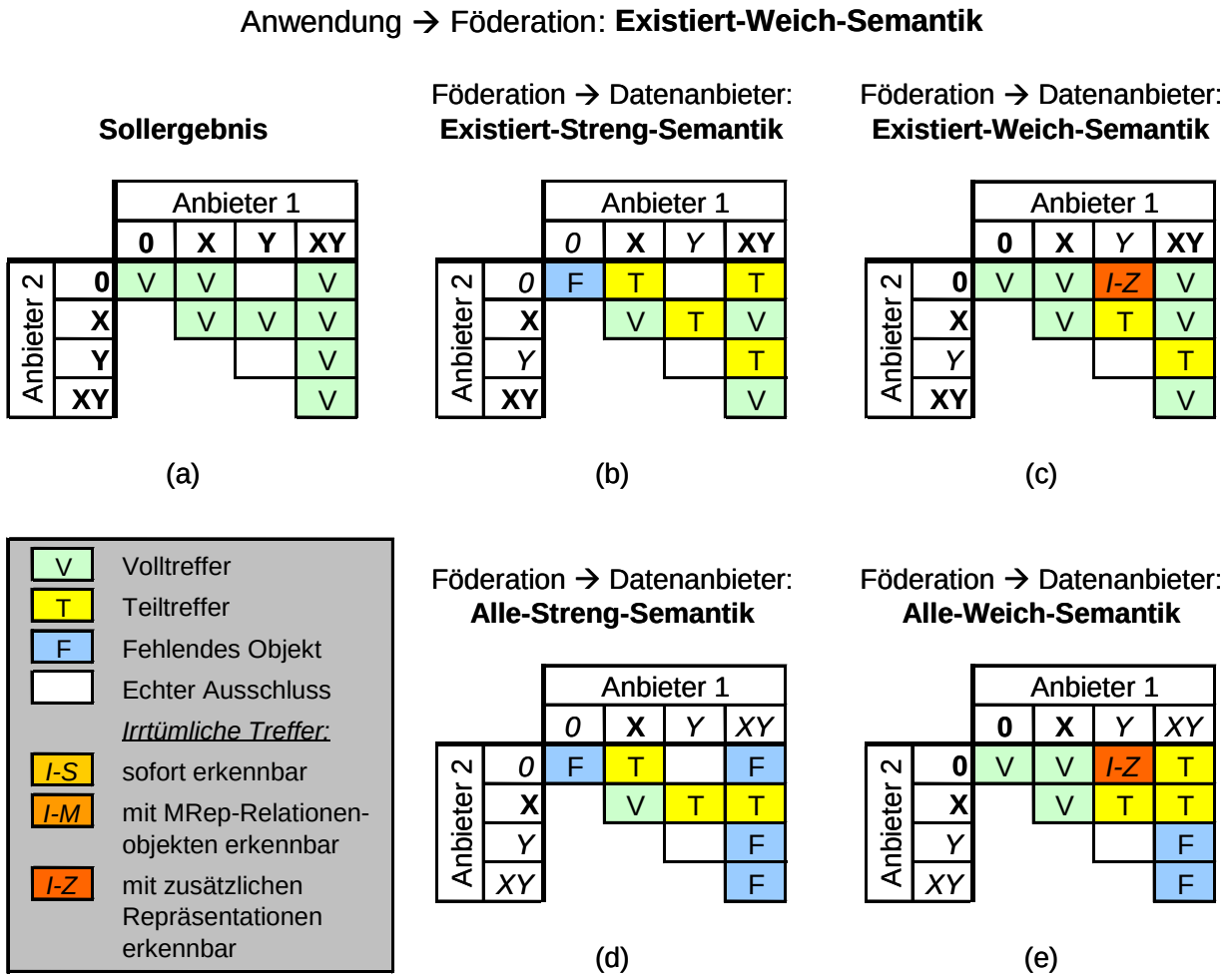


Abbildung 27: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer Anfrage mit Existiert-Weich-Semantik in der Föderation

In Abbildung 27 werden an die Föderation gerichtete E,W-Anfragen untersucht, welche die am wenigsten restriktive Anfragesemantik nutzen. Aus dem Sollergebnis kann abgelesen werden, dass sehr viele Konstellationen der Datenverteilung zu Treffern führen. Wie bei den E,S-Anfragen macht es keinen Sinn, wenn die Föderation A,S-Anfragen oder A,W-Anfragen an die Datenanbieter schickt. Zusätzlich sind auch E,S-Anfragen ungeeignet, da dann Objekte in der Ergebnismenge fehlen, deren Repräsentationen bei beiden Datenanbietern keine Werte im Attribut *B* enthalten. Die Föderation hat hier also keine Wahlmöglichkeit und muss E,W-Anfragen an die Datenanbieter schicken. Da Objekte mit fehlenden Informationen im Vergleichsattribut *B* ausdrücklich in der Ergebnismenge gewollt sind, sind nun irrtümliche Treffer erst erkennbar, nachdem die zugehörigen zusätzlichen Repräsentationen besorgt wurden.

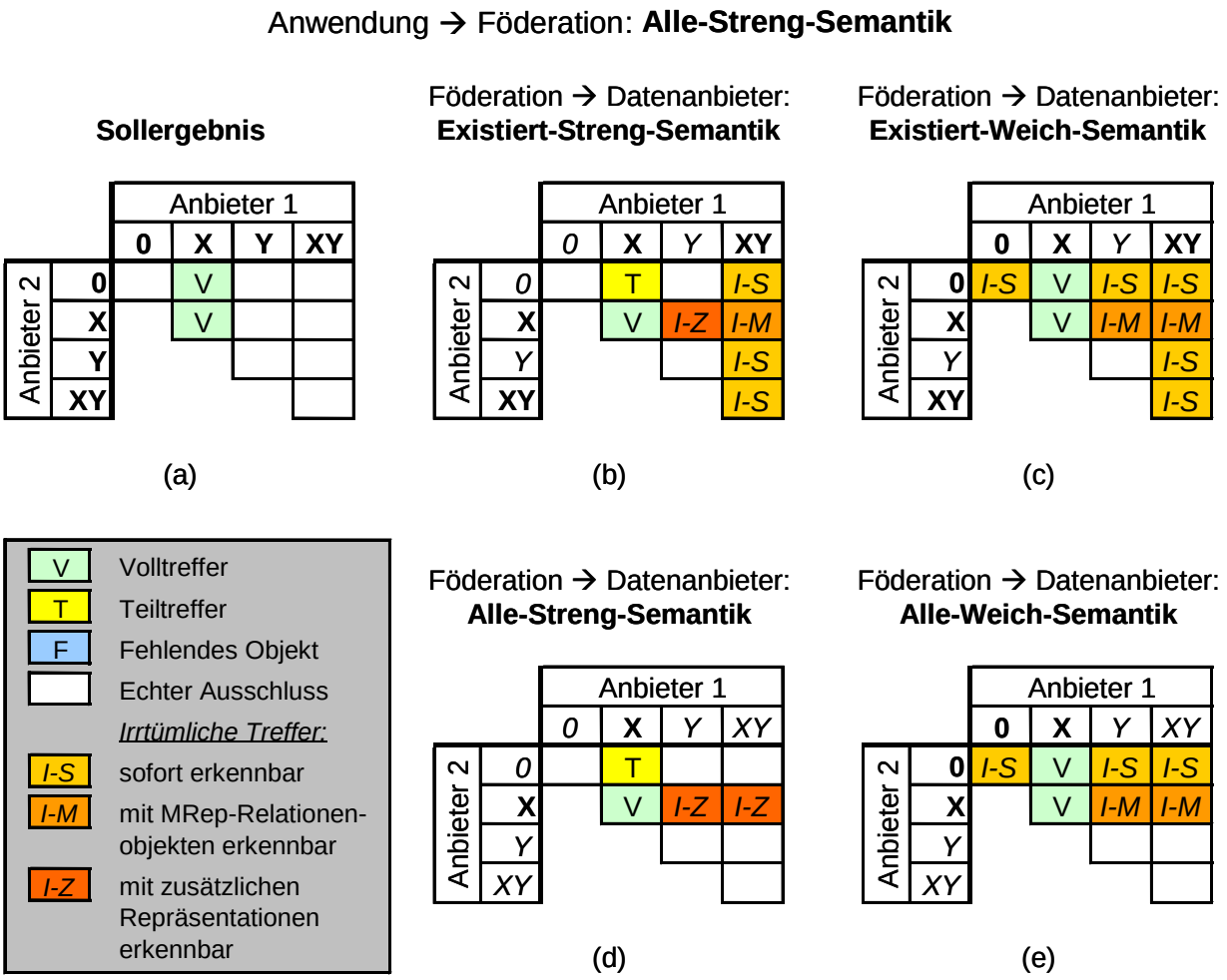


Abbildung 28: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer Anfrage mit Alle-Streng-Semantik in der Föderation

In Abbildung 28 werden an die Föderation gerichtete A,S-Anfragen untersucht, welche die restriktivste der Anfragesemantiken nutzen. Wie aus dem Sollergebnis abgelesen werden kann, führen nur sehr wenige Konstellationen der Datenverteilung zu Treffern. Hier kann die Föderation alle Semantiken bei den Anfragen an die Datenanbieter nutzen. Alle liefern ein vollständiges Ergebnis. Ebenfalls tauchen bei allen Semantiken irrtümliche Treffer auf. Ein großer Unterschied besteht zwischen den Streng- und den Weich-Semantiken. Während bei den Streng-Semantiken weniger irrtümliche Treffer übertragen werden, gibt es bei den Weich-Semantiken nur Volltreffer und keine Teiltreffer, so dass dort alle irrtümlichen Treffer schon ausgeschlossen werden können, nachdem die MRep-Relationenobjekte beschafft wurden. Wenn dort in einem MRep-Relationenobjekt ein Verweis auf eine Repräsentation enthalten ist, die nicht von einem Anbieter geliefert wurde, so kann daraus geschlossen werden, dass das zugehörige Objekt die Anfrage nicht erfüllt, da sonst alle Anbieter die entsprechenden Repräsentationen geliefert hätten. Die Existiert-Semantik ist gegenüber der Alle-Semantik im Nachteil, da sie zu mehr irrtümlichen Treffern führt ohne einen nennenswerten Vorteil zu bieten. Nach der theoretischen Abschätzung des Aufwands vom Anfang dieses Unterkapitels verarbeitet die Föderation A,S-Anfragen am besten, indem sie A,W-Anfragen an die Datenanbieter schickt.

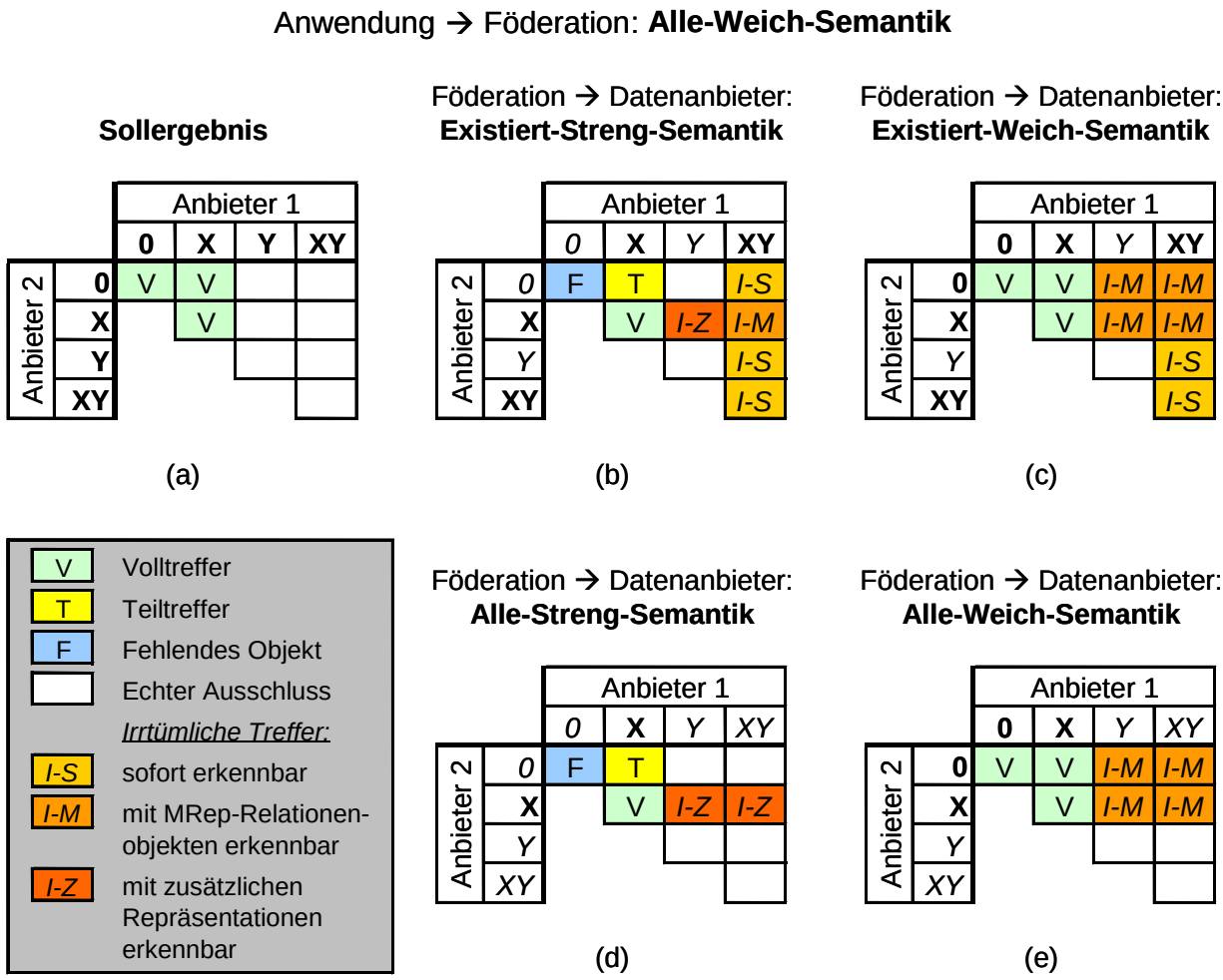


Abbildung 29: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer Anfrage mit Alle-Weich-Semantik in der Föderation

In Abbildung 29 werden an die Föderation gerichtete A,W-Anfragen untersucht. Ähnlich zu den E,W-Anfragen kann die Föderation auch hier keine Anfragen mit Streng-Semantik an die Datenanbieter schicken, da sonst die Ergebnismenge unvollständig ist. A,W-Anfragen sind gegenüber E,W-Anfragen vorzuziehen, da sie zu weniger irrtümlichen Treffern führen ohne andere Nachteile mit sich zu bringen. Zur Beantwortung von A,W-Anfragen schickt die Föderation somit am besten auch A,W-Anfragen an die Datenanbieter.

Zusammengefasst lässt sich sagen, dass die Föderation bei einer eingehenden m,n-Anfrage am besten m,W-Anfragen an die Datenanbieter schickt. Es wird also die Weich-Semantik bevorzugt und die Existiert- bzw. Alle-Semantik jeweils beibehalten.

4.4.2.3 Verarbeitung einer Konjunktion von Prädikaten bei Mehrfachrepräsentationen

In diesem Kapitel wird untersucht, welche Fälle eintreten können, wenn die Föderationskomponente Anfragen verarbeiten soll, die aus mehreren konjunktiv verknüpften Prädikaten bestehen. Hierbei wird auf den Ergebnissen des vorigen Kapitels aufgebaut, und die dort eingeführte Darstellungsweise verwendet. Anfragen mit disjunktiv verknüpften Prädikaten müssen nicht betrachtet werden, da

jeder Term der Disjunktion wie eine eigenständige Anfrage behandelt werden kann. Nach der Transformation in die disjunktive Normalform können somit beliebig komplex verknüpfte Anfragen verarbeitet werden. O. B. d. A. sei angenommen, dass die Anfrage aus zwei konjunktiv verknüpften Prädikaten bestehe, die sich auf die Attribute A und B beziehen. Für Attribut A sei der Vergleichswert U, alle weiteren möglichen Werte dieses Attributs werden durch den Wert W repräsentiert. Das Attribut B soll mit dem Wert X verglichen werden. Alle anderen Werte werden mit Y bezeichnet. Zur Abkürzung der Darstellung werden in den folgenenden Abbildungen die Werte dieser beiden Attribute durch ein Semikolon getrennt. Die Bezeichnung „0;XY“ bezieht sich also auf diejenigen Objekte bzw. Repräsentationen, bei denen das Attribut A fehlt, und die im Attribut B sowohl den gesuchten Wert X als auch einen anderen als den gesuchten Wert speichern (Y). Formal ausgedrückt werden alle Objekte gesucht, die folgende Eigenschaft haben:

$$(A = U) \wedge (B = X)$$

Eine typische Anfrage könnte nach allen Museen fragen, die eine Ausstellung zum Thema „Naturkunde“ zeigen, und die einen Hörführer (engl. audioguide) anbieten. Streng genommen besteht das Beispiel aus drei Prädikaten, da auch die Eingrenzung des Objekttyps auf Museum ein Vergleichsprädikat darstellt, bzw. sogar aus vier Prädikaten, wenn zusätzlich die Position eine Rolle spielt, beispielsweise wenn nur Museen in der Nähe des Benutzers ermittelt werden sollen. Zur Vereinfachung der Darstellung werden im Folgenden nur zwei Prädikate berücksichtigt. Wie wir sehen werden führt die Verteilung der Daten eines Objekts auf verschiedene Datenanbieter zu zusätzlichen Schwierigkeiten, wenn Anfragen aus mehr als einem Prädikat bestehen. Anfragen mit mehr als zwei Prädikaten lassen sich auf die hier diskutierten Konstellationen bei Anfragen mit zwei Prädikaten zurückführen, und müssen deshalb nicht getrennt betrachtet werden. Die Analyse wird nur für Anfragen in E,S-Semantik und A,S-Semantik detailliert durchgeführt. Die Analysen für Anfragen in den korrespondierenden Weich-Semantiken laufen in analoger Weise ab, so dass nur deren Ergebnisse präsentiert werden.

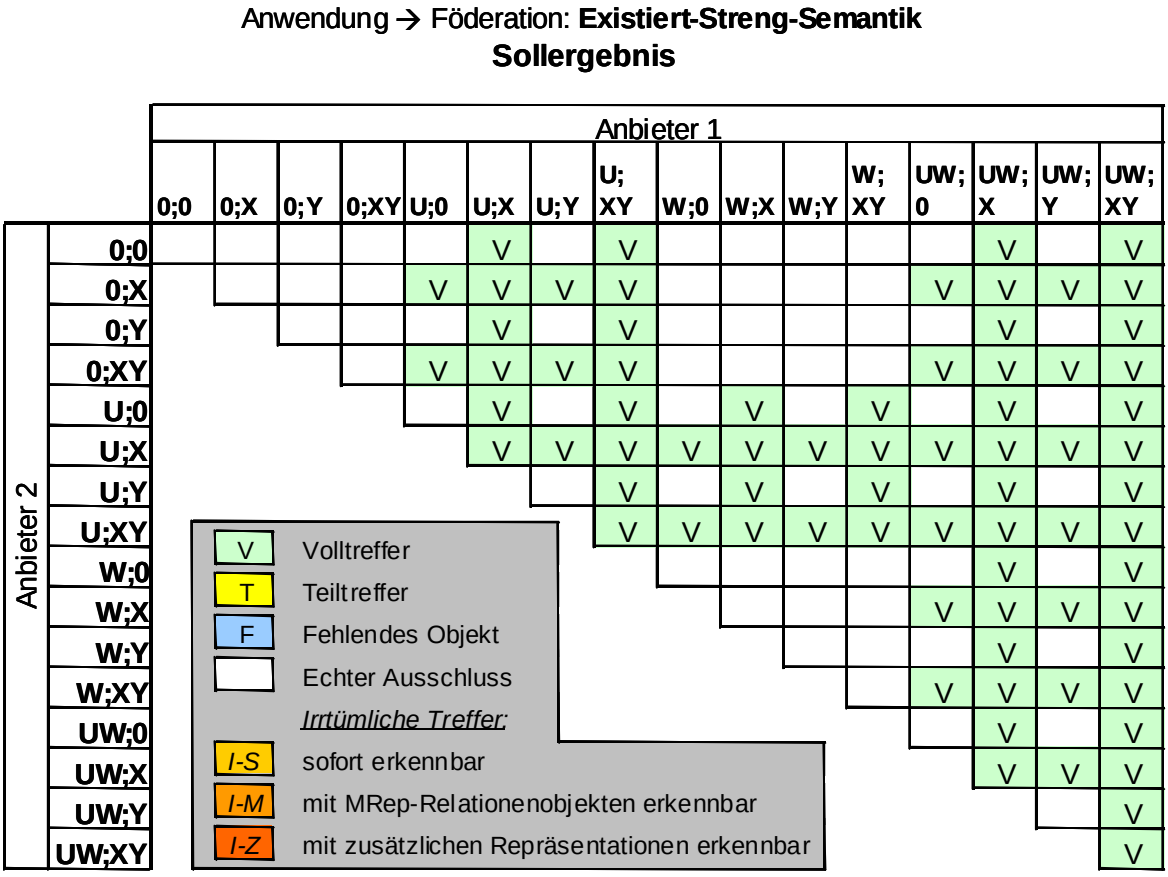


Abbildung 30: Sollergebnis einer an die Föderation gestellten Anfrage in Existiert-Streng-Semantik mit zwei konjunktiv verknüpften Prädikaten

Abbildung 30 zeigt das erwartete Sollergebnis einer Anfrage, die aus zwei konjunktiv verknüpften Prädikaten besteht, wenn die Existiert-Streng-Semantik gefordert wird. Es ist leicht zu sehen, dass bei sehr vielen der möglichen Verteilungen der Informationen des Objekts auf die Datenanbieter das Objekt die Anfrage erfüllt und in der Ergebnismenge enthalten sein soll. Dem entsprechend schwierig ist es, alle diese Fälle auch abzudecken.

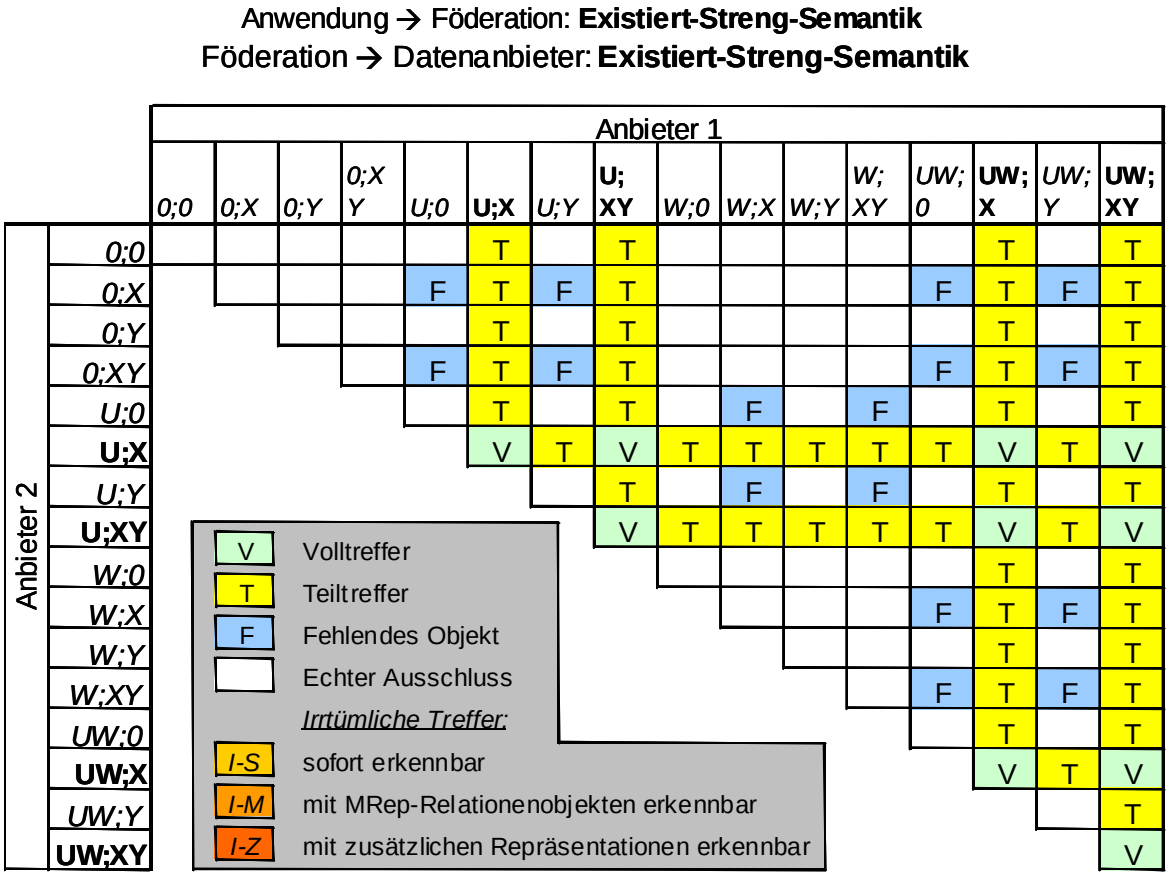


Abbildung 31: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer E,S-Anfrage mit zwei konjunktiv verknüpften Prädikaten in der Föderation, wenn E,S-Anfragen an die Datenanbieter weitergeleitet werden

In Abbildung 31 werden die Folgen dargestellt, wenn die Föderation auf eine eingehende E,S-Anfrage auch E,S-Anfragen an die Datenanbieter weiterleitet. Diese Vorgehensweise ist nicht anzuraten, da in sehr vielen Fällen Treffer nicht gefunden werden und dann in der Ergebnismenge fehlen. Auf der positiven Seite bleibt anzumerken, dass der Verarbeitungsaufwand sehr gering ist, denn es werden keine irrtümlichen Treffer an die Föderation geschickt.

Anwendung → Föderation: **Existiert-Streng-Semantik**
Föderation → Datenanbieter: **Existiert-Weich-Semantik**

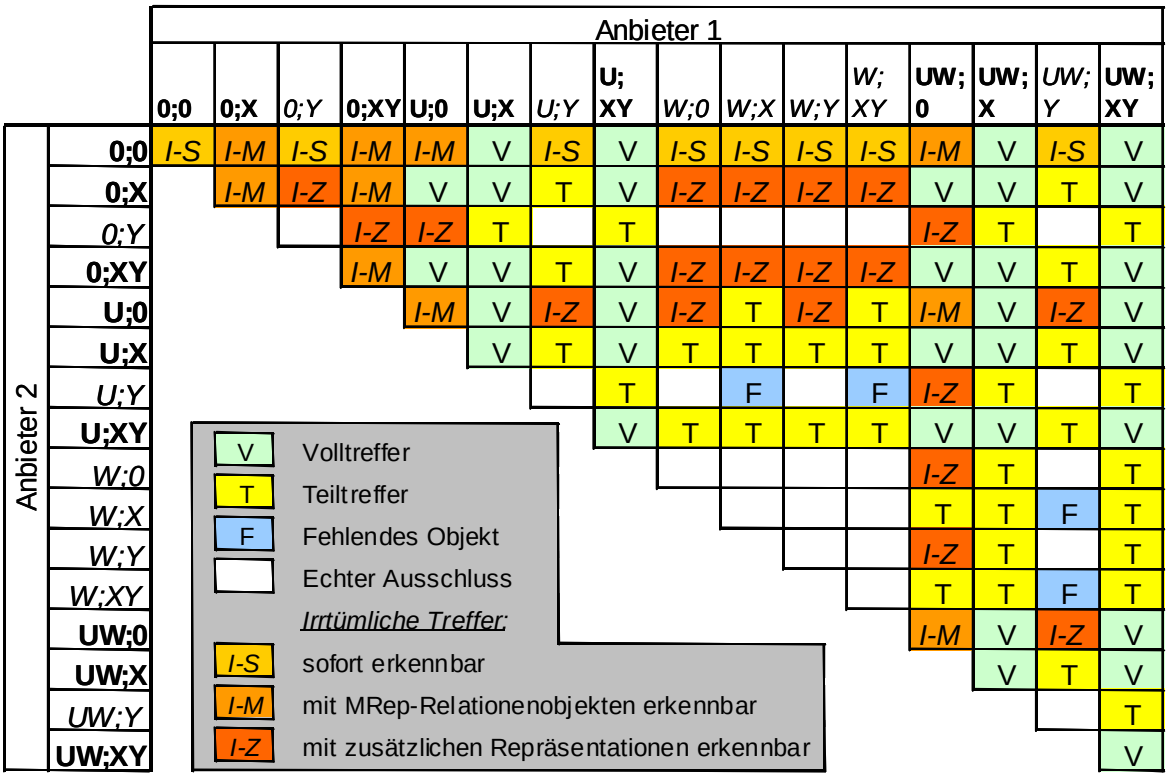


Abbildung 32: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer E,S-Anfrage mit zwei konjunktiv verknüpften Prädikaten in der Föderation, wenn E,W-Anfragen an die Datenanbieter weitergeleitet werden

In Abbildung 32 werden die verschiedenen Fälle untersucht, die bei der Verarbeitung von E,S-Anfragen in der Föderation entstehen, wenn diese E,W-Anfragen an die Datenanbieter weiterleitet. Gegenüber Abbildung 31 gibt es wesentlich weniger Fälle, die zu fehlenden Objekten führen, jedoch konnten noch nicht alle Fälle ausgemerzt werden. Die Zahl der Fälle, die zu Volltreffern führen, konnte gesteigert werden, jedoch ebenfalls die Zahl der irrtümlichen Treffer.

Anwendung → Föderation: **Existiert-Streng-Semantik**
Föderation → Datenanbieter: **Konjunktion durch Disjunktion ersetzt, E,S-Semantik**

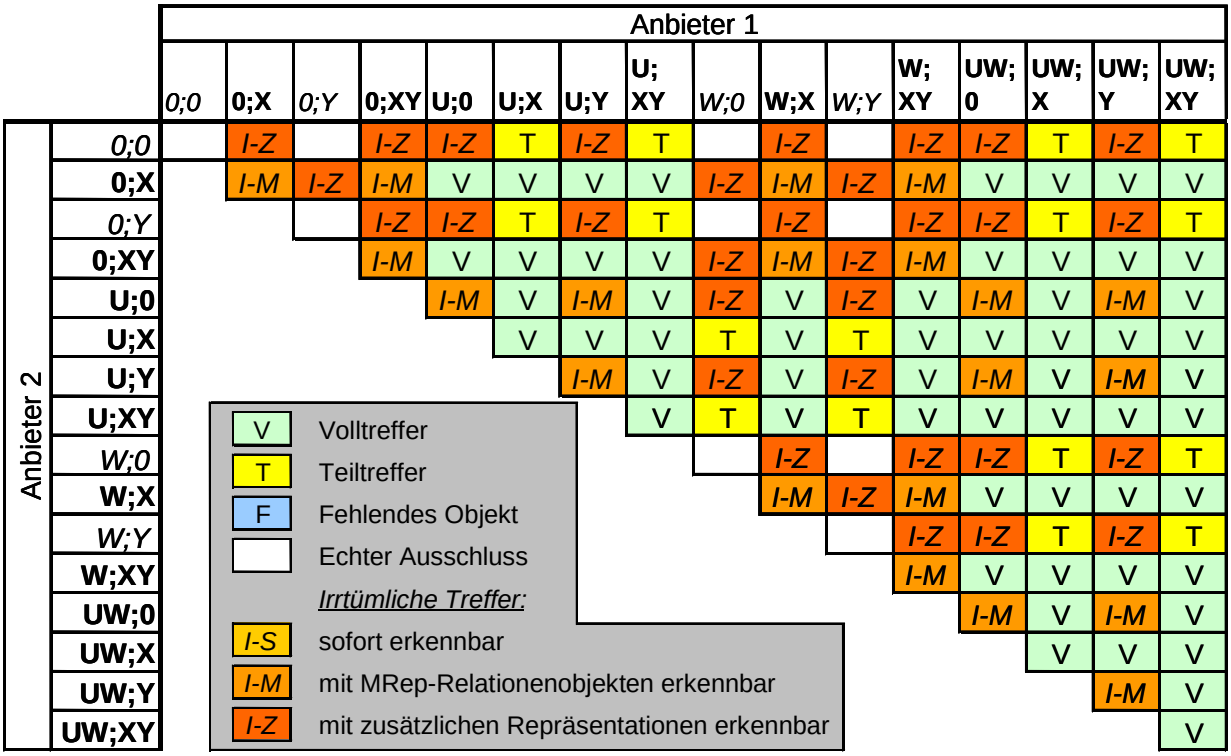


Abbildung 33: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer E,S-Anfrage mit zwei konjunktiv verknüpften Prädikaten in der Föderation, wenn E,S-Anfragen an die Datenanbieter weitergeleitet werden, bei denen die Konjunktion durch eine Disjunktion ersetzt wird

Nachdem die Existiert-Weich-Semantik die am wenigsten selektive der vorgestellten Semantiken darstellt, und es trotzdem noch zu fehlenden Objekten kommt, wenn Anfragen mit dieser Semantik an die Datenanbieter geschickt werden, muss nun nach Lösungsmöglichkeiten gesucht werden, die tiefgreifendere Änderungen vornehmen als nur die Anfragesemantik anzupassen. In Abbildung 33 werden die verschiedenen Fälle untersucht, die entstehen, wenn die Föderation eingehende E,S-Anfragen verarbeitet, indem sie die Konjunktion der konjunktiv verknüpften Prädikate durch eine Disjunktion ersetzt, und diese transformierten Anfragen an die Datenanbieter weiterleitet. Dabei wird die E,S-Semantik eingesetzt. Diese hat gegenüber der E,W-Semantik den Vorteil, dass irrtümliche Treffer wie „0;Y“ oder „W;0“ vermieden werden, die gar keinen gesuchten Wert enthalten. Mit den transformierten Anfragen ist es nun möglich alle Objekte, welche die Anfrage erfüllen, zu finden und an die Anwendung weiterzuleiten. Der Aufwand ist jedoch sehr hoch, da sehr viele irrtümliche Treffer an die Föderation geschickt werden in der Annahme, dass diese möglicherweise die Anfrage erfüllen könnten, nachdem diese durch Daten anderer zugehöriger Repräsentationen ergänzt wurden. Viele dieser irrtümlichen Treffer können erst erkannt werden, nachdem die zugehörigen zusätzlichen Repräsentationen ausgewertet wurden, welches die aufwändigste Art der Erkennung darstellt.

Eine Möglichkeit zur Begrenzung des Aufwands besteht darin, dass die Anwendung der Föderation signalisieren kann, dass sie auch mit fast vollständigen Ergeb-

nismengen zufrieden ist. Die Föderation kann damit für die Weiterleitung der Anfragen an die Datenanbieter eine andere der zuvor diskutierten Varianten wählen, die weniger aufwändig ist, aber eben auch zu mehr Lücken führt. Eine solche Qualitätsanpassung soll in dieser Arbeit jedoch nicht weiter verfolgt werden.

Die Fälle, die bei der Verarbeitung einer an die Föderation gestellten Anfrage in Existiert-Weich-Semantik auftreten, unterscheiden sich nur geringfügig von den zuvor diskutierten Fällen, die bei einer Anfrage in Existiert-Streng-Semantik auftreten. Auf eine detaillierte Untersuchung soll daher hier verzichtet werden. E,W-Anfragen erfordern ebenfalls, dass die Föderation eine konjunktive Verknüpfung der Prädikate in eine disjunktive Verknüpfung umwandelt, um eine vollständige Ergebnismenge zu erhalten. Die transformierte Anfrage wird geschickterweise in E,S-Semantik an die Datenanbieter weitergeleitet, um weniger irrtümliche Treffer zu erhalten. Zusätzlich muss die originale Anfrage in E,W-Semantik weitergeleitet werden, um den Fall abzudecken, dass beide Datenanbieter in keinem der Vergleichsattribute einen Wert vorliegen haben („0;0“ Fall).

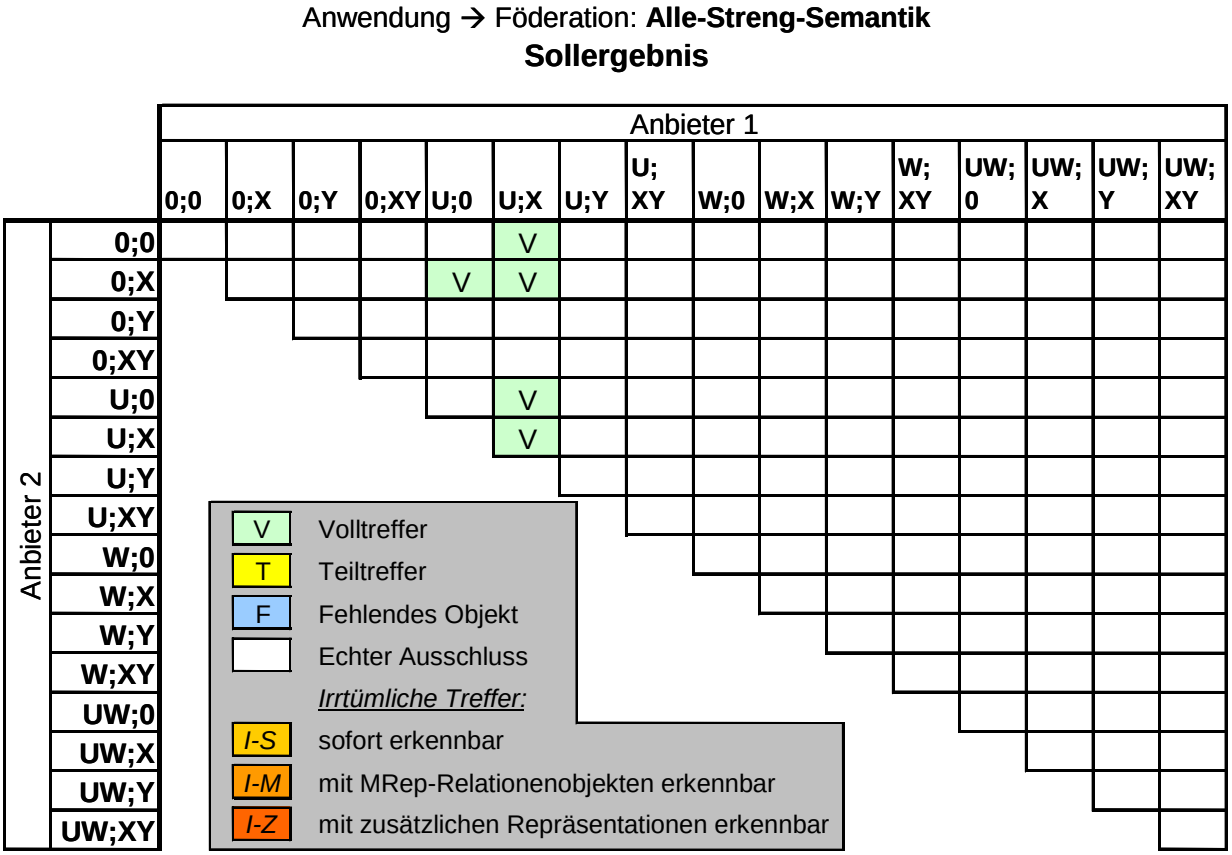


Abbildung 34: Sollergebnis einer an die Föderation gestellten Anfrage in Alle-Streng-Semantik mit zwei konjunktiv verknüpften Prädikaten

Abbildung 34 zeigt das erwartete Sollergebnis einer Anfrage, die aus zwei konjunktiv verknüpften Prädikaten besteht, wenn die Alle-Streng-Semantik gefordert wird. Diese Semantik ist sehr selektiv. Dementsprechend führen nur wenige der möglichen Verteilungen der Informationen eines Objekts auf die Datenanbieter dazu, dass das Objekt die Anfrage erfüllt. Sobald ein Objekt in einem Vergleichsattribut andere als die gesuchten Werte speichert, erfüllt es in dieser Anfragesemantik die Anfrage nicht mehr und wird von der Ergebnismenge ausgeschlossen.

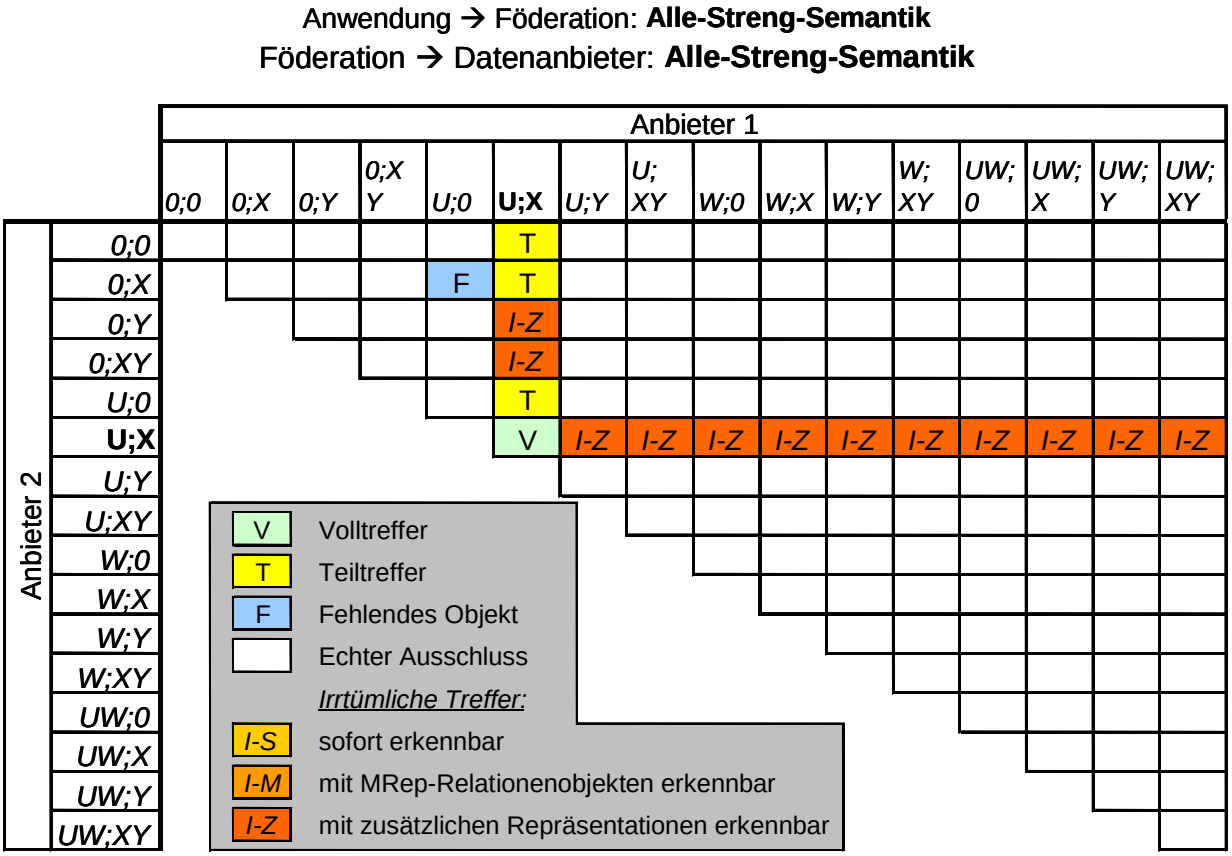


Abbildung 35: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer A,S-Anfrage mit zwei konjunktiv verknüpften Prädikaten in der Föderation, wenn A,S-Anfragen an die Datenanbieter weitergeleitet werden

In Abbildung 35 werden die Folgen dargestellt, wenn die Föderation auf eine eingehende A,S-Anfrage auch A,S-Anfragen an die Datenanbieter weiterleitet. Diese Vorgehensweise ist nur bedingt anzuraten, da zwar sehr wenige irrtümliche Treffer auftreten, aber die Ergebnismenge auch unvollständig ist. Gerade der interessante Fall, wenn die von den Prädikaten der Anfrage überprüften Informationen bei verschiedenen Datenanbietern gespeichert werden, wird nicht abgedeckt.

In Abbildung 36 werden die verschiedenen Fälle untersucht, die bei der Verarbeitung von A,S-Anfragen in der Föderation entstehen, wenn diese A,W-Anfragen an die Datenanbieter weiterleitet. Bei dieser Vorgehensweise gibt es keine fehlenden Objekte mehr. Da es auch keine Teiltreffer gibt sondern nur Volltreffer können alle irrtümlichen Treffer schon nach der Auswertung der MRep-Relationenobjekte erkannt werden. Somit werden A,S-Anfragen am besten von der Föderation verarbeitet, wenn diese A,W-Anfragen an die Datenanbieter schickt. Im Gegensatz zur Verarbeitung von Anfragen in Existiert-Semantik ist es hier nicht notwendig, die konjunktive Verknüpfung der Prädikate in eine Disjunktion zu transformieren. Dies würde nur zusätzliche irrtümliche Treffer mit sich bringen ohne weitere Vorteile zu bieten.

Die Fälle, die bei der Verarbeitung einer an die Föderation gestellten Anfrage in Alle-Weich-Semantik auftreten, unterscheiden sich nur geringfügig von den zuvor diskutierten Fällen, die bei einer Anfrage in Alle-Streng-Semantik auftreten. Auf eine detaillierte Untersuchung soll daher hier ebenfalls verzichtet werden. A,W-

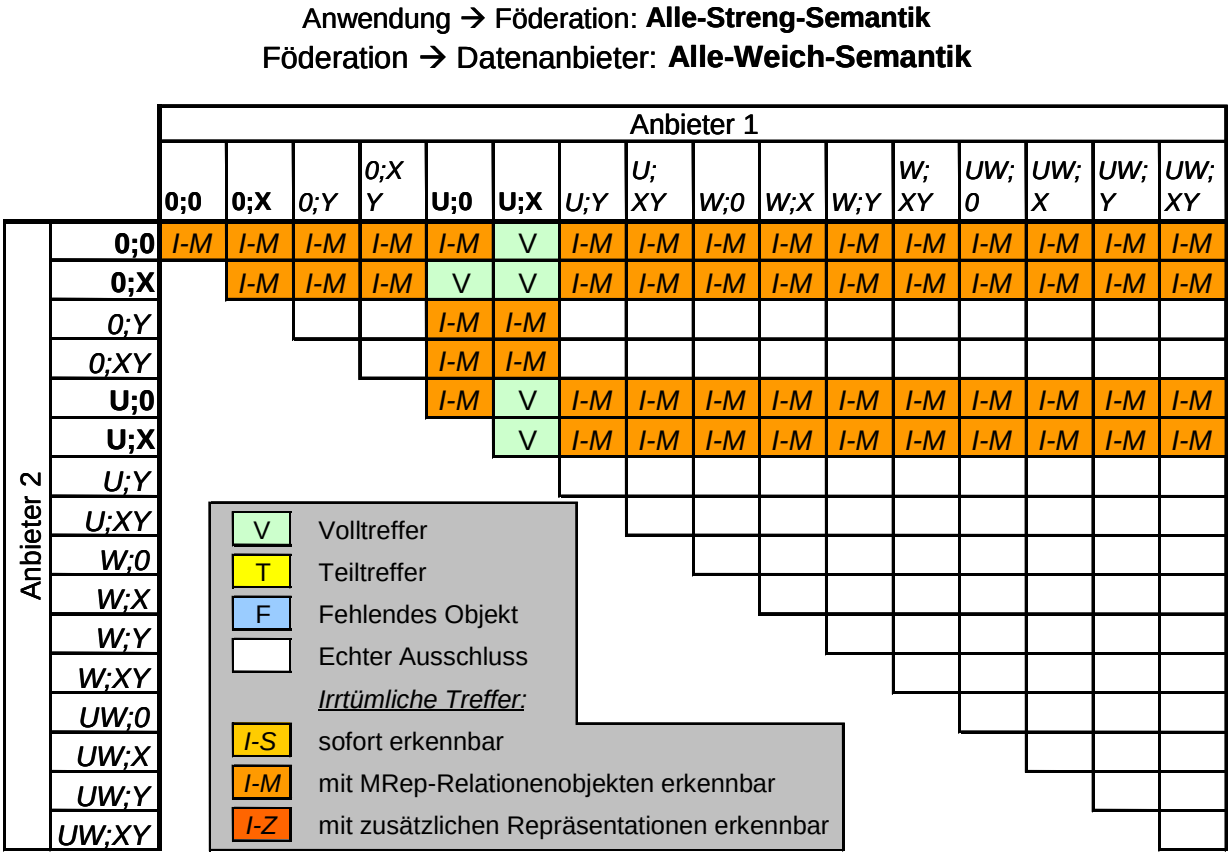


Abbildung 36: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer A,S-Anfrage mit zwei konjunktiv verknüpften Prädikaten in der Föderation, wenn A,W-Anfragen an die Datenanbieter weitergeleitet werden

Anfragen werden am besten verarbeitet, wenn die Föderation auch A,W-Anfragen an die Datenanbieter weiterleitet.

4.4.2.4 Vereinfachungen bei Widerspruchsfreiheit

Im vorigen Kapitel wurde gezeigt, dass es sehr aufwändig ist, Anfragen in Existiert-Semantik, die aus mehreren Prädikaten bestehen, zu verarbeiten, wenn eine vollständige Ergebnismenge gefordert ist. Schwierigkeiten bereiten vor allem die Fälle, in denen ein Anbieter in einem Attribut nur andere als die gesuchten Werte speichert, und somit seine lokale Repräsentation als die Anfrage nicht erfüllend einstuft, obwohl nach der Verschmelzung aller zusammengehöriger Repräsentationen das Objekt doch die Anfrage erfüllt. Das Objekt enthält also in einem Vergleichsattribut sowohl den gesuchten Wert als auch einen anderen als den gesuchten Wert.

In diesem Kapitel soll nun die Annahme getroffen werden, dass die bei verschiedenen Anbietern gespeicherten Repräsentationen eines Objekts widerspruchsfrei seien. Dies bedeutet, dass alle Repräsentationen, wenn sie einen Wert für ein Attribut speichern, entweder alle den gesuchten Wert speichern oder alle einen anderen als den gesuchten Wert speichern. Die Zahl der Fälle reduziert sich dadurch erheblich, wie auch in Abbildung 37 zu sehen ist. Fälle, die nicht auftreten können, weil

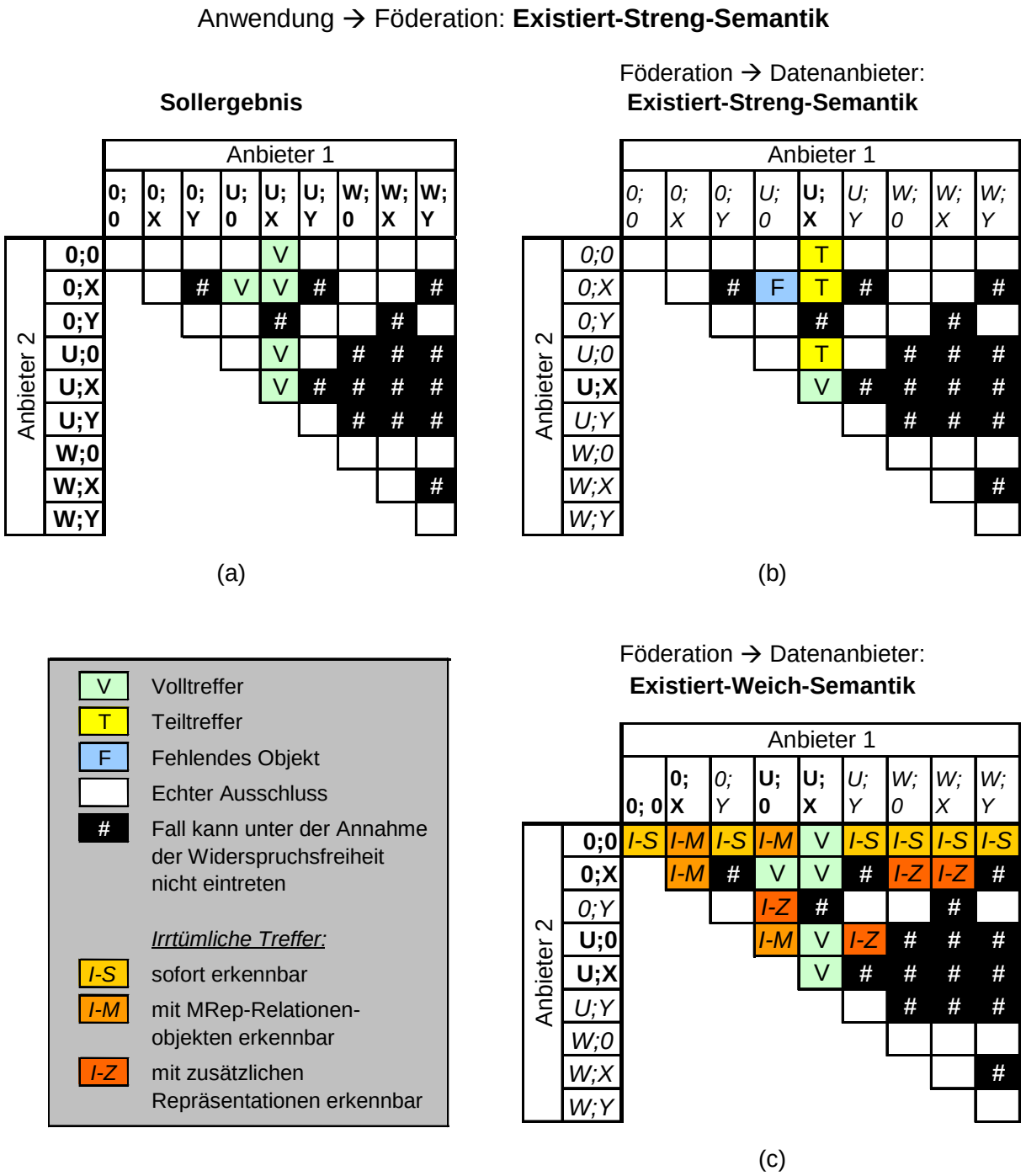


Abbildung 37: Vollständigkeit der Ergebnismenge und Aufwand bei der Verarbeitung einer Anfrage, die aus zwei konjunktiv verknüpften Prädikaten besteht, mit Existiert-Streng-Semantik in der Föderation, wenn angenommen wird, dass sich die Repräsentationen der Anbieter nicht widersprechen

dort das verschmolzene Objekt mehrere verschiedene Werte für ein Attribut enthalten würde, sind mit einer weißen Raute auf schwarzem Grund gekennzeichnet.

Unter der Annahme der Widerspruchsfreiheit gibt es keinen Unterschied zwischen der Existiert-Semantik und der Alle-Semantik, da alle Repräsentationen, wenn sie einen Wert für ein Attribut speichern, auch den selben Wert speichern. Dies wird unterstützt von der Beobachtung, dass sowohl das Sollergebnis einer solchen

Anfrage (Abbildung 37 (a)) als auch das Resultat bei der Weiterleitung von E,S-Anfragen (Abbildung 37 (b)) bzw. von E,W-Anfragen (Abbildung 37 (c)) starke Ähnlichkeiten aufweisen zu den im vorangegangenen Kapitel diskutierten Fällen bei der Verarbeitung von A,S-Anfragen (siehe Abbildungen 34 bis 36). Somit können E,S-Anfragen unter der Annahme der Widerspruchsfreiheit in der gleichen Weise verarbeitet werden wie A,S-Anfragen ohne diese Annahme. Die Verarbeitung von E,W-Anfragen vereinfacht sich unter der Annahme der Widerspruchsfreiheit in ähnlicher Weise. Auf eine detaillierte Beschreibung wird deshalb hier verzichtet.

4.4.3 Praktischer Algorithmus zur Verarbeitung von Anfragen in der Föderation

Ausgehend von den Analysen der vorigen Kapitel soll in diesem Kapitel die praktische Umsetzung der theoretischen Erkenntnisse vorgestellt werden. Gleichzeitig stellt dies eine Zusammenfassung der vorigen Kapitel dar. Die Föderation führt bei der Verarbeitung von Anfragen den in Abbildung 38 dargestellten Algorithmus aus. Zur Wiederholung sei hier erwähnt, dass in diesem Kapitel nur der Selektionsteil einer Anfrage betrachtet wird und die restlichen Teile wie Projektion oder Schemaauswahl hier ausgeblendet werden. Dementsprechend bezieht sich der Begriff Anfrage in diesem Kapitel auf den Selektionsteil einer Anfrage. Der Begriff Anfrageverarbeitung meint verkürzend die Verarbeitung des Selektionsteils einer Anfrage.

Der von der Föderation ausgeführte Algorithmus zur Verarbeitung von Anfragen gliedert sich in mehrere Schritte, die im Folgenden näher erläutert werden:

- **Negationen zu den Blättern verlagern**: Mit Hilfe der De Morganschen Gesetze können Negationen in den Verknüpfungen der Prädikate einer Anfrage bis direkt zu den Prädikaten (den Blättern des Verknüpfungsbaums) verlagert werden. Dies dient dazu, die in Kapitel 4.1.2 in Satz 4.13 beschriebene Inversion der Anfragesemantik bei einem negierten Objektprädikat zu vermeiden.
- **DNF bilden**: Die disjunktive Normalform (DNF) des Verknüpfungsbaums kann durch anwenden der Distributiv- und Assoziativgesetze gebildet werden. Dies vereinfacht sowohl die weitere Verarbeitung der Anfrage als auch die Erklärung davon. Beim Abgleich vorhandener Cache-Inhalte mit einer gegebenen Anfrage muss diese in DNF vorliegen, siehe Kapitel 5.5.
- **Anfragesemantik anpassen**: Je nach der Semantik der an die Föderation gestellten Anfrage und der Anzahl der Prädikate müssen die an die Datenanbieter geschickten Anfragen teilweise eine abweichende Semantik verwenden, siehe Abbildung 39. Bei Anfragen in Existiert-Streng-Semantik (E,S-Semantik) mit zwei oder mehr Prädikaten muss eine transformierte Anfrage erstellt werden, bei der alle einzelnen Prädikate (unabhängig von deren Verknüpfung in der gegebenen Anfrage) disjunktiv verknüpft sind. Die transformierte Anfrage wird in E,S-Semantik an die Datenanbieter geschickt. Bei Anfragen in Existiert-Weich-Semantik (E,W-Semantik) mit zwei oder mehr Prädikaten muss dieselbe transformierte Anfrage erstellt werden und in E,S-Semantik an die Datenanbieter geschickt werden. Zusätzlich muss auch die gegebene Anfrage in E,W-Semantik an die Datenanbieter geschickt werden.

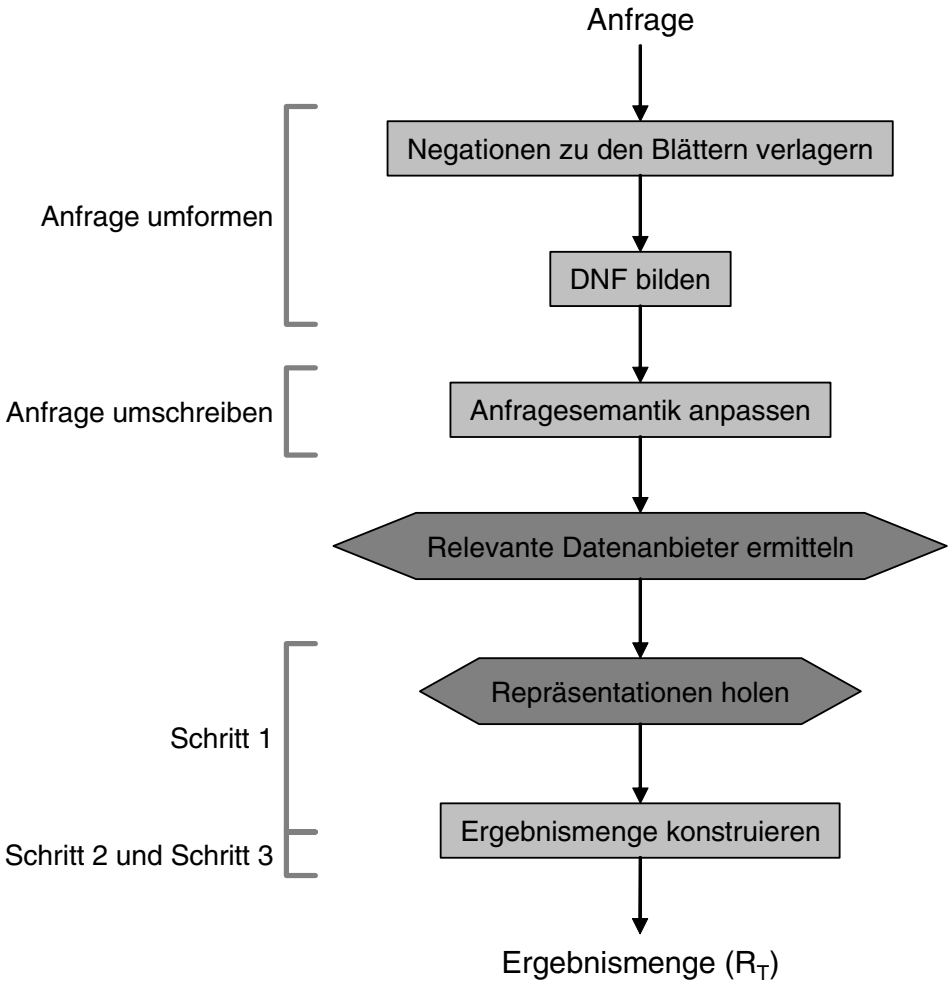


Abbildung 38: Algorithmus zur Verarbeitung von Anfragen in der Föderation. Die Schritte 1 bis 3 beziehen sich auf den in Kapitel 4.4.2.2 beschriebenen geschätzten Aufwand.

Semantik der an die Föderation gestellten Anfrage Q	E,S		E,W		A,S		A,W	
Anzahl der Prädikate	1	2+	1	2+	1	2+	1	2+
Semantik der an die Datenanbieter weitergeleiteten Anfragen	E,W	$Q_{transformiert}$: E,S	E,W	$Q_{transformiert}$: E,S + Q: E,W	A,W	A,W	A,W	A,W

Abbildung 39: Bestimmung der Semantik der Anfragen, die an die relevanten Datenanbieter weitergeleitet werden, in Abhängigkeit von der Semantik der gegebenen Anfrage.

Dieser Schritt folgt aus den in den Kapiteln 4.4.2.2 und 4.4.2.3 durchgeführten Analysen.

- **Relevante Datenanbieter ermitteln**: Um die für eine gegebene Anfrage relevanten Datenanbieter zu ermitteln wird eine spezielle Anfrage an das räumliche Verzeichnis gestellt, bei dem sich alle Datenanbieter registrieren müssen. Als Ergebnis wird eine Liste der relevanten Datenanbieter zurückgeliefert. Die Relevanz eines Anbieters wird derzeit mittels der Objekttypen der gespeicherten Objekte und des geographischen Gebiets, in dem

sich die Objekte befinden können (Dienstgebiet), ermittelt. Prädikate auf anderen Attributen werden bei der Bestimmung der relevanten Datenanbieter derzeit nicht berücksichtigt. Diese Vorgehensweise ist angepasst an die Eigenschaften der Anwendungsdomäne der kontextbezogenen Anwendungen, in welcher am häufigsten nach dem Objekttyp und nach der geographischen Position selektiert wird.

- **Repräsentationen holen** und **Ergebnismenge konstruieren**: Die letzten beiden Schritte gehören zusammen und stellen diejenigen Operationen dar, die tatsächlich mit Nutzdaten umgehen. Zunächst werden die angepassten Anfragen an die relevanten Datenanbieter geschickt und deren Ergebnismengen (R_1 ... R_n) mit den auf die Anfrage passenden Repräsentationen geholt. Ausgehend von diesen initialen Ergebnismengen wird in dem in Abbildung 25 skizzierten dreischrittigen Prozess die finale Ergebnismenge konstruiert. Der zugehörige Algorithmus ist in Abbildung 40 dargestellt. Die drei Schritte entsprechen den in Kapitel 4.4.2.2 beschriebenen Schritten zur Abschätzung des Aufwands der dort erörterten Fallunterscheidung je nach Verteilung der Daten eines Objekts auf Datenanbieter. In jedem Schritt werden zusätzliche Informationen hinzugeholt, so dass jeder Schritt eine Kommunikation mit anderen Systemen erfordert, was jeweils mit zusätzlichen Latenzzeiten verbunden ist. Je nach Verteilung der Daten auf die Datenanbieter kann die Verarbeitung auch schon nach ein oder zwei Schritten abgeschlossen sein.

In Abbildung 40 ist der Ablauf des Algorithmus zur Konstruktion der Ergebnismenge dargestellt. Der Algorithmus hat als Eingabe die initialen Ergebnismengen R_1 bis R_n der relevanten Datenanbieter. Die Ausgabe des Algorithmus ist die finale Ergebnismenge R_T, in der die Repräsentationen so weit wie möglich zu Objekten verschmolzen wurden und fehlende Informationen vervollständigt worden sind. Nicht alle Repräsentationen aus den initialen Ergebnismengen landen jedoch in der finalen Ergebnismenge. Diese landen entweder in der Menge R_L der zwischengelagerten Repräsentationen, die dort auf zugehörige Repräsentationen warten, mit denen zusammen sie die Anfrage erfüllen können, oder sie landen in der Menge R_N der Nicht-Treffer, die alle Repräsentationen enthält, die auf gar keinen Fall die Anfrage erfüllen können. Des Weiteren gibt es die Menge R_U der unvollständigen Repräsentationen und die Menge R_V der vollständigen Repräsentationen. Während R_U Repräsentationen enthält, zu denen es zugehörige Repräsentationen gibt, die noch nicht in der Föderation vorliegen, enthält R_V Repräsentationen, zu denen alle zugehörigen Repräsentationen vorliegen und welche zu einem sogenannten Objekt verschmolzen worden sind. Die Menge R_S enthält (noch unvollständige) Repräsentationen, die auf jeden Fall im Endergebnis enthalten sein werden. Die Menge R_M enthält Repräsentationen, die möglicherweise im Endergebnis enthalten sein werden, je nachdem welche Daten in den zugehörigen Repräsentationen gespeichert sind. Alle Mengen sind lokal und stellen den Datenfluss zwischen zwei Blöcken im Ablaufdiagramm dar. Lediglich die Mengen R_T und R_N sind global.

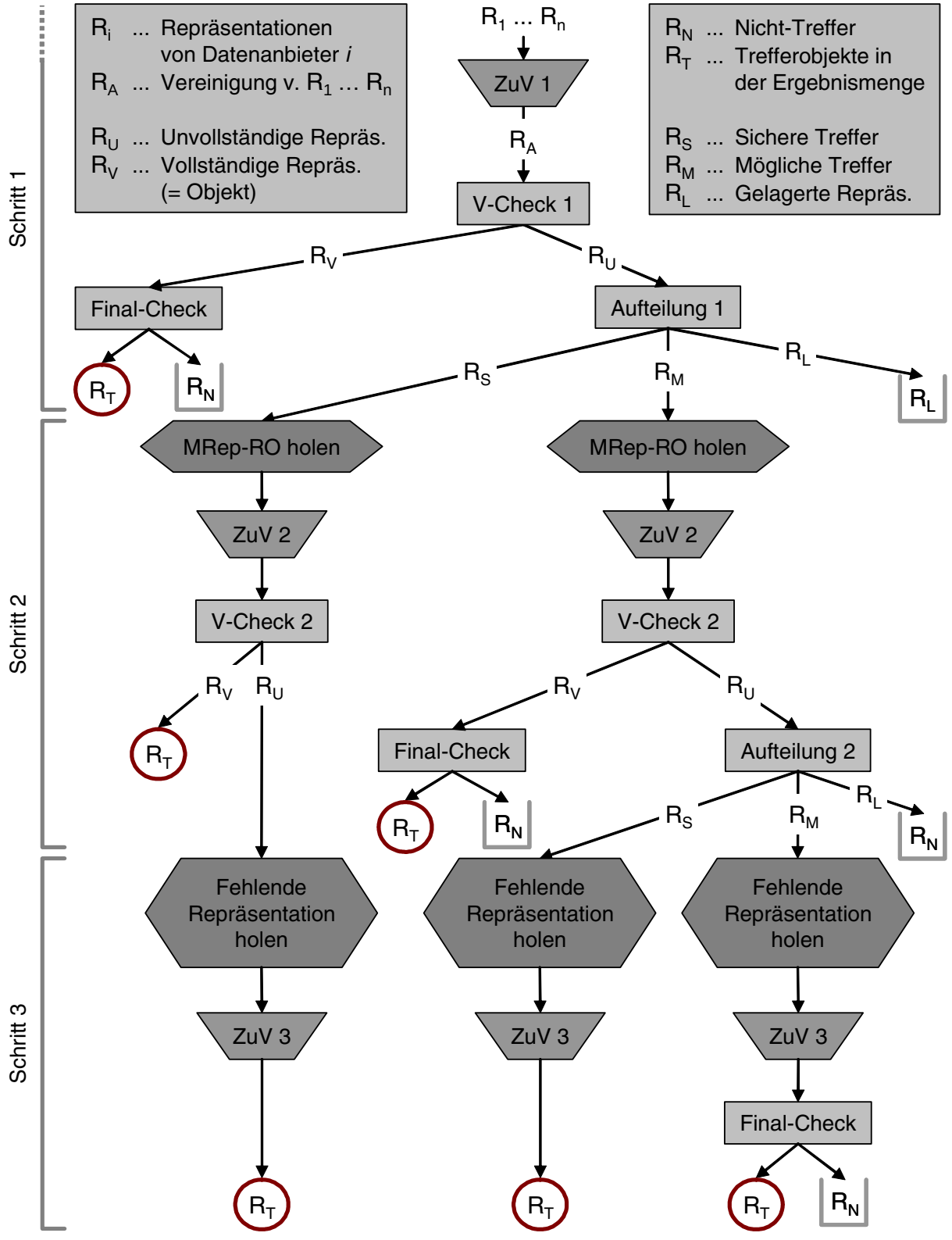


Abbildung 40: Algorithmus zur Konstruktion der Ergebnismenge bei der Verarbeitung von Anfragen in der Föderation. Die Schritte 1 bis 3 beziehen sich auf die Aufwandsschätzungen in Kapitel 4.4.2.2.

Auch normale Objekte, die aus nur einer Repräsentation bestehen, müssen diesen Algorithmus durchlaufen, da zu Beginn des Algorithmus die Repräsentation eines

normalen Objekts nicht von einer Mehrfachrepräsentation unterschieden werden kann. Dies kann erst festgestellt werden, wenn es zu einer Repräsentation keine MRep-Relationenobjekte gibt.

Im Folgenden werden die Verarbeitungsblöcke des in Abbildung 40 dargestellten Ablaufdiagramms näher erläutert:

- **ZuV 1** (Zuordnen und Verschmelzen 1)
  Eingabe: R_1 ... R_N – initiale Ergebnismengen der Datenanbieter
  Ausgabe: R_A – Vereinigung der initialen Ergebnismengen

  Dieser Block ist optional. Es werden zusammengehörige Repräsentationen verschmolzen, soweit dies mit den vorhandenen Informationen möglich ist. Zur Zuordnung der Repräsentationen können entweder identische Objekt-IDs oder domänenspezifische Verfahren eingesetzt werden, siehe Kapitel 4.4.1.1. Zur Verschmelzung können beide der in Kapitel 4.4.1.2 vorgestellten Ansätze eingesetzt werden. Wenn dieser Block ausgelassen wird, so werden im nachfolgenden Block V-Check 1 alle Repräsentationen als unvollständig eingestuft.

- **V-Check 1** (Vollständigkeits-Check 1)
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_U – Menge der unvollständigen Repräsentationen
  R_V – Menge der vollständigen Repräsentationen (Objekte)

  In diesem Block wird für jede Repräsentation der Eingabemenge überprüft, ob alle zugehörigen Repräsentationen auch in der Eingabemenge vorhanden sind bzw. in dieser Repräsentation durch eine vorangegangene Verschmelzung enthalten sind. Konkret wird hier geprüft ob jeder der befragten Datenanbieter, der potentiell die Repräsentation speichern könnte, eine zugehörige Repräsentation geliefert hat. Wenn dies der Fall ist, ist die Repräsentation vollständig, sonst ist sie unvollständig. Ein Datenanbieter kann eine Repräsentation potentiell speichern, wenn diese in dessen Dienstgebiet liegt und deren Typ zu den Objekttypen gehört, die der Datenanbieter speichert. Für viele Repräsentationen und insbesondere für viele einzelne Repräsentationen bedeutet dies, dass sie als unvollständig eingestuft werden.

- **Final-Check**
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_T – Menge der Treffer
  R_N – Menge der Nicht-Treffer

  Dieser Block führt für jedes Objekt in der Eingabemenge eine abschließende Überprüfung mit der gegebenen Anfrage durch. Erfüllt das Objekt die Anfrage landet es in der finalen Ergebnismenge, ansonsten landet es im Ausschusstopf. Alle eingehenden Objekte müssen vollständig sein, d. h. alle zugehörigen Repräsentationen müssen hier schon verschmolzen worden sein.

- **Aufteilung 1**
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_S – Menge der noch unvollständigen aber sicheren Treffer
  R_M – Menge der möglichen Treffer
  R_L – Menge der vorerst nicht weiter aktiv verfolgten Repräsentationen

  Dieser Block überprüft für jede Repräsentation der Eingabemenge ob sie die gegebene Anfrage Q erfüllt. Wenn eine Repräsentation ein zu vergleichendes Attribut nicht enthält, so wird dem Ergebnis des entsprechenden Objektprädikats der besondere Wert *null* zugeordnet, wie dies auch schon in [Codd86] vorgeschlagen wird. Die Verknüpfung mehrerer Objektprädikate erfolgt entsprechend der in Abbildung 41 dargestellten dreiwertigen Logik. Für jede Repräsentation kann so bestimmt werden, ob sie die gegebene Anfrage erfüllt oder nicht. Abhängig von der Semantik der gegebenen Anfrage und der Anzahl der darin enthaltenen Prädikate kann in Abbildung 42 die Ausgabemenge R_S, R_M oder R_L für die Repräsentation abgelesen werden. Die Tabelle folgt aus den in den Kapiteln 4.4.2.2 und 4.4.2.3 gesammelten Erkenntnissen.

∧	**wahr**	**null**	**falsch**
wahr	wahr	null	falsch
null	null	null	falsch
falsch	falsch	falsch	falsch

∨	**wahr**	**null**	**falsch**
wahr	wahr	wahr	wahr
null	wahr	null	null
falsch	wahr	null	falsch

	¬
wahr	falsch
null	null
falsch	wahr

Abbildung 41: Boolesche Verknüpfungen in dreiwertiger Logik [Codd86].

Semantik		E,S		E,W		A,S		A,W	
Anzahl der Prädikate		1	2+	1	2+	1	2+	1	2+
Ergebnis der Auswertung von Q	null	R_L	R_M	R_M	R_M	R_L	R_M	R_M	R_M
	wahr	R_S	R_S	R_S	R_S	R_M	R_M	R_M	R_M
	falsch	R_L	R_M	R_L	R_M	R_L	R_L	R_L	R_L

Menge der
R_S ... Sicheren Treffer
R_M ... Möglichen Treffer
R_L ... Zwischengelagerten Repräsentationen

Abbildung 42: Bestimmung der Zielmenge im Schritt „Aufteilung 1“.

- **MRep-RO holen** (MRep-Relationenobjekte holen)
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_A – Menge von Repräsentationen mit zugehörigen MRep-Relationen-
  objekten

  In diesem Block werden zu allen Repräsentationen in der Eingabemenge die zugehörigen MRep-Relationenobjekte geholt. Diese können entweder bei einem Datenanbieter gespeichert sein, der auch die Repräsentation speichert, oder bei einem unabhängigen Drittanbieter. Der Speicherort des MRep-Relationenobjekts kann entweder direkt über sogenannte Rückwärtsverweise, die in der Repräsentation enthalten sind, oder mittels einer weiteren föderierten Anfrage nach diesen Relationenobjekten ermittelt werden. Die föderierte Anfrage ist zwingend für Repräsentationen, die keine Rückwärtsverweise enthalten. Hierbei

kann ausgenutzt werden, dass auch Relationenobjekte über eine Positionsangabe verfügen [ADD+05].

- **ZuV 2** (Zuordnen und Verschmelzen 2)
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_A – Menge von verschmolzenen Repräsentationen

  In diesem Block werden zusammengehörige Repräsentationen verschmolzen. Die Zuordnung erfolgt hierbei mittels der zuvor geholten MRep-Relationenobjekte, siehe Kapitel 4.4.1.1. Zur Verschmelzung können beide der in Kapitel 4.4.1.2 vorgestellten Ansätze eingesetzt werden.

- **V-Check 2** (Vollständigkeits-Check 2)
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_U – Menge der unvollständigen Repräsentationen
  R_V – Menge der vollständigen Repräsentationen (Objekte)

  In diesem Block wird für jede Repräsentation der Eingabemenge überprüft, ob sie schon mit allen Repräsentationen, auf die in den zugehörigen MRep-Relationenobjekten verwiesen wird, verschmolzen worden ist. Wenn dies der Fall ist, ist die Repräsentation vollständig, also ein Objekt, und landet in der Menge R_V. Andernfalls landet sie in der Menge R_U. Falls zu einer Repräsentation kein zugehöriges MRep-Relationenobjekt gefunden werden konnte, dann ist dies ein normales Objekt, das alleine schon vollständig ist, und landet in der Menge R_V.

- **Aufteilung 2**
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_S – Menge der noch unvollständigen aber sicheren Treffer
  R_M – Menge der möglichen Treffer
  R_L – Menge der vorerst nicht weiter aktiv verfolgten Repräsentationen

  Dieser Block sortiert jede Repräsentation aus der Eingabemenge in eine der drei Ausgabemengen R_S, R_M oder R_L ein. Jede Repräsentation durchläuft dabei zwei Prüfungen: die Vorprüfung (Abbildung 43) und die Hauptprüfung (Abbildung 44). Die Vorprüfung wird nur durchgeführt, wenn die gegebene Anfrage eine der Existiert-Semantiken nutzt und mindestens zwei Prädikate enthält. Ansonsten wird gleich die Hauptprüfung durchgeführt. Die Tabellen folgen aus den in den Kapiteln 4.4.2.2 und 4.4.2.3 gesammelten Erkenntnissen.

Semantik				E,S	E,W
Anzahl der Prädikate				2+	2+
Ergebnis der Auswertung von Q	null	Ergebnis der Auswertung von $Q_{transformiert}$	null	R_L	weiter zur Hauptprüfung
			wahr		R_M
			falsch		R_L
	wahr			R_S	R_S
	falsch			weiter zur Hauptprüfung	weiter zur Hauptprüfung

Abbildung 43: Vorprüfung zur Bestimmung der Zielmenge im Schritt „Aufteilung 2“.

Bei der Vorprüfung wird nochmals geprüft, ob eine (inzwischen mit anderen Repräsentationen verschmolzene) Repräsentation die gegebene Anfrage Q erfüllt. Bei Anfragen in Existiert-Streng-Semantik muss unter Umständen zusätzlich überprüft werden, ob die Repräsentation die transformierte Anfrage $Q_{transformiert}$ erfüllt. Je nach Ergebnis der Prüfung kann entweder direkt schon die Ausgabemenge bestimmt werden, oder die Repräsentation wird zur Hauptprüfung weitergeleitet, siehe Abbildung 43.

Semantik	E,S		E,W		A,S		A,W	
Anzahl der Prädikate	1	2+	1	2+	1	2+	1	2+
Alle fehlenden Repräsentationen liegen bei Datenanbietern, die schon gefragt wurden	kommt nicht vor	R_L	R_L	R_L	R_L	R_L	R_L	R_L
Es gibt fehlende Repräsentationen, die bei Datenanbietern liegen, die noch nicht gefragt wurden	kommt nicht vor	R_M	R_M	R_M	R_M	R_M	R_M	R_M

Abbildung 44: Hauptprüfung zur Bestimmung der Zielmenge im Schritt „Aufteilung 2“.

Der Hauptprüfung müssen sich alle unvollständigen Repräsentationen unterziehen, bei denen die Möglichkeit besteht, dass diese durch zusätzliche zugehörige Repräsentationen, die andere als die gesuchten Werte enthalten, noch disqualifiziert werden können. Zum einen ist dies der Fall, wenn eine Repräsentation in einem Vergleichsattribut noch keinen Wert hat (NULL Wert). Zum anderen ist dies bei beiden Alle-Semantiken möglich. Für jede Repräsentation in der Eingabemenge wird deshalb geschaut, wo sich die fehlenden Repräsentationen befinden, auf welche die zugehörigen MRep-Relationenobjekte verweisen, siehe Abbildung 44. Wenn die noch fehlenden Repräsentationen alle bei Datenanbietern liegen, die schon gefragt wurden, so kann daraus geschlossen werden, dass diese Repräsentationen nur abträgliche Informationen enthalten können, da sonst die jeweilige Repräsentation schon in der initialen Antwort des Datenanbieters enthalten gewesen wäre. Folglich kann die Repräsentation disqualifiziert werden. Wenn es fehlende Repräsentationen gibt, die bei Datenanbietern liegen, die noch nicht gefragt wurden, so kann keine Annahme über deren Informationen getroffen werden. Zur abschließenden Bewertung der Repräsentation müssen die Daten der fehlenden Repräsentationen auf jeden Fall geholt werden und zu einem Objekt verschmolzen werden.

- **Fehlende Repräsentationen holen**
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_A – Menge von Repräsentationen (R_E + geholte Repräsentationen)

  In diesem Block werden für jede Repräsentation der Eingabemenge diejenigen Repräsentation von den jeweiligen Datenanbietern geholt, auf die in den zugehörigen MRep-Relationenobjekten verwiesen wird, und die noch nicht in der Föderation vorliegen (in der Eingabemenge dieses Blocks oder in der Menge der zwischengelagerten Repräsentationen R_L). Der Speicherort einer Repräsentation kann direkt aus dem Verweis abgeleitet werden.

- **ZuV 3** (Zuordnen und Verschmelzen 3)
  Eingabe: R_E – Menge von Repräsentationen
  Ausgabe: R_A – Menge von verschmolzenen Repräsentationen

  Siehe ZuV 2.

Der vorstehend beschriebene Algorithmus liefert für alle Anfragesemantiken eine vollständige Ergebnismenge, wenn die Zuordnungen zwischen Mehrfachrepräsentationen in MRep-Relationenobjekten gespeichert sind. Fehlen diese MRep-Relationenobjekte, so können nur diejenigen Repräsentationen zusammengeführt werden, die in den initialen Ergebnismengen der Datenanbieter enthalten sind. In den Kapiteln 4.4.2.2 und 4.4.2.3 sind dies die Fälle, die zu einem Volltreffer führen. Die Ergebnismenge kann dann unvollständige Objekte, Mehrfachrepräsentationen bzw. Duplikate und irrtümliche Treffer enthalten.

Ein normales Objekt, das aus nur einer einzigen Repräsentation besteht, durchläuft den Algorithmus wie folgt. Im Schritt „ZuV 1“ werden keine weiteren zugehörigen Repräsentationen gefunden. Somit wird das Objekt im „V-Check 1“ in die Menge R_U der unvollständigen Repräsentationen eingruppiert, da an dieser Stelle in der Regel noch nicht ausgeschlossen werden kann, dass es bei anderen Datenanbietern noch zugehörige Repräsentationen gibt. Falls das Objekt im Schritt „Aufteilung 1“ in die Menge R_S der sicheren Treffer eingruppiert wird, dann wird anschließend versucht, zugehörige MRep-Relationenobjekte zu finden. Da es bei einem normalen Objekt keine gibt, kann das Objekt nach dem Schritt „V-Check 2“ in die Ergebnismenge überführt werden. Der Ablauf ist ähnlich, wenn das Objekt im Schritt „Aufteilung 1“ in die Menge R_M der möglichen Treffer eingruppiert wird. Nur folgt hier nach dem „V-Check 2“ der „Final-Check“, welcher das Objekt als Treffer oder Nicht-Treffer bewertet. Noch schneller verworfen wird das Objekt, wenn es in „Aufteilung 1“ in die Menge R_L verschoben wird. Repräsentationen in dieser Menge werden nicht weiter aktiv verfolgt. In jedem Fall hat die Verarbeitung von Mehrfachrepräsentationen zur Folge, dass sich die Verarbeitung normaler Objekte verlangsamt, da erst nachdem keine zugehörigen MRep-Relationenobjekte gefunden worden sind feststeht, dass es sicht um ein normales Objekt handelt.

4.4.4 Abgrenzung

Einige Arbeiten im Bereich der Datenbanken haben sich schon mit der Verarbeitung von fehlenden Informationen beschäftigt [ImLi84, Codd86, Codd87, Gess90]. Hierbei wurde das Konzept der NULL Werte zur Repräsentation der fehlenden Informationen sowie eine dreiwertige Logik für deren Verarbeitung bei der Auswertung von Prädikaten vorgeschlagen. Einige Arbeiten unterscheiden zwischen fehlender, weil unbekannter, und fehlender, weil nicht anwendbarer, Information. Diese setzen eine vierwertige Logik bei der Verarbeitung ein. Andere Arbeiten verallgemeinern dies weiter und definierten weitere Semantiken für fehlende Informationen, so dass dort n-wertige Logik eingesetzt wird. In der Praxis hat sich die einfachere Lösung mit nur einem NULL Wert durchgesetzt, weshalb dieses Konzept sowie die dreiwertige Logik hier übernommen wird. Die Arbeiten gehen jedoch weder auf die Speicherung mehrerer Werte für ein Attribut (sogenannte Mehrfachattribute) noch auf die Probleme, die durch die verteilte Speicherung der Daten entstehen, ein.

Morrissey beschäftigt sich in [Morr90] mit der Verarbeitung von unsicherer Information. Ein Attribut kann dort entweder einen präzisen Wert oder eine Reihe von möglichen Werten haben. Diese können entweder in Form eines Wertebereichs (p-range) oder in Form einer Liste (p-domain) angegeben werden. Der dezidierte Wert „not known“ (nk) gehört ebenfalls zur Kategorie der möglichen Werte und ist mit den hier verwendeten NULL Werten vergleichbar. Während dort p-domains mögliche alternative Werte eines Attributs kodieren, können die Werte eines Mehrfachattributs alle gleichzeitig zutreffen. Obwohl die Arbeit ein anderes Ziel verfolgt, nämlich unscharfe Daten zu verarbeiten und die Ergebnisse nach dem Grad der Unsicherheit zu sortieren, sind doch einige Ergebnisse auf das hier diskutierte Problem übertragbar, wenn man die dortigen p-domains als einen Sonderfall der Mehrfachattribute ansieht. Das Ergebnis einer Anfrage besteht bei Morrissey aus zwei Mengen: der Menge der Objekte, von denen sicher ist, dass sie auf jeden Fall die Anfrage erfüllen, und der Menge der Objekte, die möglicherweise die Anfrage erfüllen können, wenn eine p-domain oder ein p-range durch den richtigen präzisen Wert ersetzt werden würde. Interessanterweise entspricht die erste Ergebnismenge dem Ergebnis einer Anfrage in Alle-Streng-Semantik und die zweite Ergebnismenge dem Ergebnis einer Anfrage in Existiert-Weich-Semantik. Demnach stellt diese Arbeit hier eine Verallgemeinerung der Ergebnisse von Morrissey dar, die diese um die zugehörigen komplementären Semantiken, um die Definition eines inversen Objektprädikats durch Inversion der Semantik, sowie um die Betrachtung der Problematik bei der verteilten Speicherung der Daten erweitert. Umgekehrt lässt sich daraus schließen, dass Mehrfachattribute zusammen mit den hier erarbeiteten Anfragesemantiken und deren föderierter Verarbeitung die Grundlage bilden, auf der die Föderation um die Verarbeitung von unscharfen oder unsicheren Daten erweitert werden kann.

Ein ähnlicher Ansatz wird in [INV91] vorgestellt. Dort wird die Repräsentation sogenannter möglicher Welten durch das Konzept der ODER-Objekte ermöglicht. Diese Objekte können in einem Attribut mehrere mögliche Werte speichern. Anfragen werden als Kombination aus einer prädikatenlogischen Formel und einem Quantor definiert. Der Quantor gibt dabei an, ob die Formel für eine mögliche Instanz des Objekts oder für alle möglichen Instanzen des Objekts erfüllt sein muss, damit das Objekt als Treffer gewertet wird. Die Anfragen dort sind somit vergleichbar mit Anfragen in Existiert-Semantik bzw. in Alle-Semantik. Es werden jedoch keine fehlenden Informationen im Sinne von NULL Werten betrachtet. Probleme, die durch die verteilte Speicherung der Daten entstehen, sind ebenfalls nicht Gegenstand der Betrachtungen dort.

Das Problem der Verarbeitung konjunktiver Anfragen, wenn die Daten eines Objekts verteilt sind, wurde in ähnlicher Weise im Rahmen des Garlic Projekts [Fagi96] diskutiert. Dort soll eine föderierte Multimedia-Datenbank erstellt werden, bei der jedes Attribut eines Objekts in einem speziellen Quellsystem gespeichert wird. Als Beispiel wird eine CD-Datenbank betrachtet, bei der Informationen wie Künstler oder Albumname in einer klassischen Datenbank verwaltet werden und ein Bild des Einbands in der speziellen QBIC Datenbank (query by image content) gespeichert werden, welche insbesondere eine Ähnlichkeitssuche nach Bildinhalten ermöglicht. Es werden konjunktive Anfragen wie (Künstler=“Beatles“ und Einband=“rot“) betrachtet, was insofern ähnlich zum hier betrachteten Problem ist, dass die Informationen zu den beiden konjunktiv verknüpften Prädikaten auf verschiedenen Systemen lagern. Anders als hier wird dort jedoch davon ausgegangen,

dass jedes Quellsystem nur einen Teilaspekt jedes Objekts verwaltet aber dafür Informationen für jedes Objekt bereithält. Dem entsprechend verarbeitet jedes Quellsystem nur ein Prädikat der Anfrage. Dies ist vergleichbar mit der in Kapitel 4.5.3.3 diskutierten Transformation der Anfragen, bei der die Konjunktion durch eine Disjunktion ersetzt wird. Hier wird diese Transformation nur durchgeführt, wo sie unvermeidbar ist, da dadurch viele irrtümliche Treffer verursacht werden, wohingegen sie bei Garlic stets durchgeführt wird. Fehlende Werte eines Objektattributs (NULL Werte) sowie Mehrfachattribute werden bei Garlic nicht diskutiert. Der Fokus in [Fagi96] liegt auf der Verarbeitung von Ähnlichkeitsanfragen, die zu unscharfen (engl. fuzzy) Vergleichsergebnissen führen. Die Ergebnisse sollen sortiert nach ihrer Ähnlichkeit zur Anfrage ausgegeben werden und die Ergebnismenge ist typischerweise auf die besten k Ergebnisse begrenzt. Der Garlic-Ansatz ist also nicht direkt auf das hier diskutierte Problem übertragbar.

Im Bereich der Datenfusion wurden unter anderem spezielle Join-Operatoren [YaÖz99, GPZ01], spezielle Aggregationsfunktionen [SCS00, WaZa00] oder das FUSE BY Statement [BlNa05] vorgeschlagen. Datenbereinigungssysteme [GFSS00, RaHe01] eliminieren Duplikate und Konflikte in einem zentralen Datenbestand. Spezielle Datenfusionssysteme [SAB+95, PAG96, GPQ+97, MoAn04] erlauben es, die Daten einer festen vorher bekannten Menge von Datenanbietern zusammenzuführen und unterstützen dies durch spezielle Mechanismen wie z. B. Fusionsregeln. In allen Fällen steht die Bereinigung von Inkonsistenzen in einer vorgegebenen Tupel- bzw. Objektmenge im Vordergrund. Die Auswahl der zusammenzuführenden Objekte wird, anders als hier, nicht als Problem betrachtet, da die Auswahl entweder über andere als die zu verschmelzenden Attribute erfolgt, oder ganze Datensätze verschmolzen werden und das Ergebnis materialisiert wird.

Das Aurora Projekt verfolgt einen Mediator-basierten Datenintegrationsansatz, bei dem die Daten erst zur Laufzeit auf Instanzebene integriert werden und Konflikte in den Daten explizit in den Anfragen toleriert werden können [YaÖz99]. Als Prädikatevaluierungsstrategien werden „HighConfidence“ und „PossibleAtAll“ diskutiert, was der Alle- bzw. der Existiert-Semantik hier entspricht. Jedoch dürfen Datenfragmente (vergleichbar mit Datenanbietern) keine NULL Werte enthalten, wodurch die Anfrageverarbeitung vereinfacht wird. Die Basisanfrageverarbeitungsstrategie dort sieht vor, zunächst alle Daten von allen Quellfragmenten zu holen und diese mit einem speziellen Merge-Join zu verschmelzen. Nur in wenigen speziellen Fällen werden die Selektionsprädikate direkt auf den Quellfragmenten ausgewertet bevor der Merge-Join berechnet wird. Dem gegenüber wird hier immer angestrebt, eine möglichst selektive Anfrage an die Datenanbieter zu schicken. Der interessante Fall, wenn die Datenanbieter jeweils in einem Vergleichsattribut den gesuchten Wert und in einem anderen Vergleichsattribut einen anderen als den gesuchten Wert speichern, so dass nur durch die Kombination der Daten verschiedener Anbieter ein Objekt als die Anfrage erfüllend eingestuft werden kann, wird gar nicht angesprochen. Die Zuordnung der Repräsentationen verschiedener Anbieter erfolgt über eine eindeutige Kennung, dort PID genannt. Dies erfordert unter Umständen, dass die Kennungen aller bisher als Treffer identifizierten Objekte an alle Datenanbieter geschickt werden müssen, um zusätzliche zugehörige Repräsentationen zu beschaffen, um damit die Qualifikation eines Objekts abschließend beurteilen zu können. Dem gegenüber werden zur Zuordnung hier MRep-Relationenobjekte eingesetzt, welche entweder direkt per Rückwärtsverweis oder indirekt über deren Position gefunden werden können. Darüber hinaus wird hier

die Semantik der an die Datenanbieter geschickten Anfragen angepasst, um das zu übertragende Datenvolumen in den folgenden Schritten zu minimieren, und um eine vollständige Ergebnismenge zu erhalten, was ohne diese Anpassung nicht immer gewährleistet ist.

4.4.5 Zusammenfassung

In diesem Kapitel wurde untersucht, welcher zusätzliche Aufwand bei der Verarbeitung von Anfragen nötig ist, damit Mehrfachrepräsentationen korrekt behandelt werden können, insbesondere wenn die für die Objektauswahl relevanten Daten auf mehrere Anbieter verteilt sind. In der Ergebnismenge sollen Mehrfachrepräsentationen nur noch als verschmolzene Objekte auftauchen, die alle zugehörigen Repräsentationen enthalten. Hierfür wurde zunächst das Konzept der Mehrfachrepräsentationen vorgestellt. Anschließend wurde eine systematische theoretische Analyse aller möglichen Verteilungen der für die Anfrage relevanten Daten auf mehrere Anbieter durchgeführt. Aus dem Ergebnis der Analyse konnte ein praktischer Algorithmus zur Verarbeitung der Anfragen abgeleitet werden. Die Diskussion der verwandten Arbeiten hat gezeigt, dass einige Ansätze ähnliche Anfragesemantiken verwenden, jedoch keiner davon die durch die verteilte Speicherung der Mehrfachrepräsentationen hervorgerufenen Probleme beleuchtet.

Die Analysen in diesem Kapitel haben gezeigt, dass die Behandlung von Mehrfachrepräsentationen einiges an Zusatzaufwand erfordert, vor allem, wenn die Korrektheit und Vollständigkeit der Ergebnismenge garantiert werden sollen. Der Zusatzaufwand entsteht erstens durch die Anpassung der Semantik der weitergeleiteten Anfragen, wodurch einige Repräsentationen von den Datenanbietern geholt werden, die in der Föderationskomponente verworfen werden. Zweitens müssen unter Umständen MRep-Relationenobjekte und fehlende Repräsentationen in zwei weiteren Schritten von den Datenanbietern geholt werden. Schließlich führt bei der Existiert-Semantik die Anfragetransformation in disjunktiv verknüpfte Einzelprädikate, welche für ein vollständiges Ergebnis erforderlich ist, zu noch höherem Zusatzaufwand. Dies ist besonders ungünstig, weil die Existiert-Streng-Semantik die intuitive Semantik ist, welche auch am häufigsten genutzt wird. Des Weiteren kann der Algorithmus nur dann eine vollständige und korrekte Ergebnismenge liefern, wenn alle Mehrfachrepräsentationen über MRep-Relationenobjekte zugeordnet werden können. Lediglich bei Volltreffern führen auch andere Zuordnungsmethoden zum Erfolg. Der vorgestellte Algorithmus führt auch bei normalen Objekten zu Mehraufwand, da vorab nicht ausgeschlossen werden kann, dass es sich dabei um Mehrfachrepräsentationen handelt.

Wenn die Anforderungen gelockert werden, kann die Verarbeitung erheblich beschleunigt werden, indem nur noch der erste Schritt des Algorithmus ausgeführt wird. Dann werden nur noch Repräsentationen verschmolzen, die von den Datenanbietern auf die weitergeleitete Anfrage hin geliefert werden. MRep-Relationenobjekte werden dabei nicht ausgewertet. Die Ergebnismenge kann dann unvollständige Objekte, Duplikate und irrtümliche Treffer enthalten, wobei der zurückgelieferte Teil eines irrtümlichen Treffers immerhin tatsächlich die Anfrage erfüllt. Ähnliche Folgen hat das Fehlen von MRep-Relationenobjekten. Auch dann können nicht mehr alle Repräsentationen zugeordnet werden.

Zur Beschleunigung der Anfrageverarbeitung sind einige Optimierungen denkbar. Bei der Existiert-Semantik können sichere Trefferobjekte schon frühzeitig an die Anwendung ausgeliefert werden. Falls weitere zugehörige Repräsentationen gefunden werden müssen die schon gelieferten Objekte nachträglich ergänzt werden. Des Weiteren kann die von der Anwendung geschickte Anfrage umgeschrieben werden, so dass gleich alle potentiell benötigten MRep-Relationenobjekte mitgeliefert werden. Hierbei kann die Position des MRep-Relationenobjekts ausgenutzt werden. Ein offenes Problem ist dabei, wie erreicht werden kann, dass möglichst nur die MRep-Relationenobjekte geliefert werden, die zu den restlichen angefragten Objekten gehören, und nicht noch viel zu viele unerwünschte MRep-Relationenobjekte geliefert werden.

Schließlich kann der Detektionsprozess für Mehrfachrepräsentationen automatisiert werden. Zum einen kann ein nachgeschalteter Prozess die Anfrageergebnisse analysieren nachdem diese an die Anwendung ausgeliefert wurden. Zum anderen kann ein Hintergrundprozess wie ein Web Crawler die Datenbestände der Anbieter nach Mehrfachrepräsentationen systematisch durchforsten. Des Weiteren können auch die Ergebnisse der domänenspezifischen Zuordnungsverfahren in einer Art Cache zwischengespeichert werden oder bei einem Datenanbieter materialisiert werden.

4.5 Weitere Maßnahmen

Die in diesem Kapitel beschriebenen weiteren Maßnahmen sind Ideen auf konzeptioneller Ebene zur Weiterentwicklung der Föderationskomponente.

Obwohl durch die Simplizität der Anfragesprache wenig Raum für Optimierungen bleibt, gibt es noch weitere Maßnahmen, welche die Effizienz der Verarbeitung föderierter Anfragen steigern können. Insbesondere soll hier die Arbeitsteilung zwischen Föderationsknoten angesprochen werden. Sobald eine Föderationskomponente die Ergebnisse bisheriger Anfragen zwischenspeichern kann und zur Beantwortung zukünftiger Anfragen heranziehen kann, wie in Kapitel 5.5 angedeutet wird, können Synergieeffekte erzielt werden, wenn sich mehrere Föderationskomponenten koordinieren. Die Föderationskomponenten können Zuständigkeitsgebiete absprechen, so dass sie in ihrem Cache einen Großteil der Daten ihres Zuständigkeitsbereichs vorhalten können, und damit die Weiterleitung der meisten Anfragen an die Datenanbieter entfallen kann. Falls eine Anwendung nach Daten fragt, die nicht in den Zuständigkeitsbereich fallen, so kann die Föderationskomponente bei der zuständigen Nachbarföderationskomponente die Daten besorgen, anstatt bei dem Datenanbieter selbst. Dies ist insbesondere dann sinnvoll, wenn die Föderationskomponenten untereinander besser vernetzt sind als mit dem Datenanbieter, und wenn die Auslastung der Föderationskomponenten diese Zusatzbelastung erlaubt.

Die Trefferquote im Cache kann weiter erhöht werden, wenn eine Föderationskomponente für alle mobilen Benutzer zuständig ist, die mit einer bestimmten Basistation des Mobilfunknetzes verbunden sind, und der Zuständigkeitsbereich entsprechend definiert wird [DSB+04]. Diese Benutzer stellen typischerweise Anfragen nach ähnlichen Daten, die in der Umgebung der Basisstation relevant sind, so dass die Anfragen einen hohen Überlappungsgrad aufweisen.

Schließlich kann dieser Cache-Ansatz auch auf die Ergebnisse von Ergebnisaufbereitungsoperatoren aus Kapitel 5.4.2 ausgedehnt werden. So könnte eine Föderationskomponente sich auf das Zwischenspeichern von Ergebnissen des Generalisierungsoperators spezialisieren und eine andere auf die Ergebnisse des Straßendatenverschmelzungsoperators. Aufrufe des entsprechenden Operators in der einen Komponente können dann auf den Cache der anderen Komponente umgeleitet werden, so dass der Operator nicht erneut ausgeführt werden muss. Der hiermit eingesparte Aufwand sollte den zum Koordinieren der Föderationskomponenten benötigten Aufwand wesentlich übersteigen, so dass damit die Leistungsfähigkeit des Gesamtsystems gesteigert werden kann. Die Zusammenarbeit mehrerer Föderationskomponenten bietet ein enormes Potential.

Gemeinsam können die Föderationskomponenten einen größeren Ausschnitt des globalen Umgebungsmodells abdecken und somit sowohl die Datenanbieter entlasten als auch die Anfrageverarbeitung beschleunigen. In gleicher Weise können sie eine größere Menge der Ergebnisse von Ergebnisaufbereitungsoperatoren zwischenspeichern, und somit die wiederholte aufwändige Verarbeitung der immer gleichen Daten reduzieren.

4.6 Bewertung

In diesem Kapitel wurde die föderierte Anfrageverarbeitung hinsichtlich der Interaktionen zwischen den verschiedenen Komponenten des Nexus-Systems erläutert. Die vorgestellten Verfahren berücksichtigen die *Autonomie* der Datenanbieter indem sie die relevanten Datenanbieter mittels des Verzeichnisdienstes bestimmen. Sie integrieren die Daten der relevanten Anbieter (*Integration*) und machen damit die Verteilung der Daten transparent für die Anwendungen (*Transparenz*). Für jeden der beiden *Anfragetypen* wurde ein entsprechendes Verarbeitungsverfahren vorgestellt, das jeweils das bestmögliche mit den aktuell verfügbaren Datenanbietern mögliche Ergebnis produziert (*Informationsmaximierung*). Bei der Verarbeitung von föderierten Nachbarschaftsanfragen und durch die Zusammenarbeit mehrerer Föderationskomponenten wurde die *Effizienz* der Anfrageverarbeitung optimiert.

Durch die niedrigen Anforderungen der Anwendungen in der Domäne der kontextbezogenen Anwendungen an die Mächtigkeit der Anfragesprache, mit der Daten aus dem Umgebungsmodell abgefragt werden sollen, konnte die Verarbeitung von Gebietsanfragen sehr einfach und hart verdrahtet gestaltet werden. Föderierte Nachbarschaftsanfragen können am effizientesten verarbeitet werden, wenn Statistiken über die globale Objektdichte vorliegen, und wenn die Anfragen mit moderater Parallelität an die Datenanbieter weitergeleitet werden. Selbst wenn die Datenanbieter nur Gebietsanfragen verarbeiten können, können damit trotzdem föderierte Nachbarschaftsanfragen emuliert werden, bei einem geringfügig höheren Aufwand als wenn die Anbieter selbst Nachbarschaftsanfragen verarbeiten können. Die Behandlung von Mehrfachrepräsentationen erfordert einigen Zusatzaufwand in der Anfrageverarbeitung, wenn sichergestellt werden soll, dass alle bekannten Mehrfachrepräsentationen zusammengeführt werden und dass die Ergebnismenge keine irrtümlichen Treffer enthält. Es wurde analysiert, welche Auswirkungen die Änderung der Semantik der an die Datenanbieter weitergeleiteten Anfragen auf die Vollständigkeit und Korrektheit des Ergebnisses hat. Aus den Ergebnissen wurde

ein Mehrfachrepräsentationen berücksichtigender Algorithmus zur Verarbeitung von Anfragen abgeleitet. Die Kooperation mehrerer Föderationskomponenten, insbesondere durch Absprachen bezüglich der Cacheinhalte, stellt eine vielversprechende Maßnahme zur Steigerung der Effizienz der Anfrageverarbeitung dar.

Durch Einbringen von Wissen aus der Anwendungsdomäne und durch Ausnutzen ihrer Spezifika wie beispielsweise des Orts- bzw. Kontextbezugs der Anfragen konnten die Verfahren einfacher und effizienter gestaltet werden. Dies spiegelt sich insbesondere in der Bestimmung der relevanten Datenanbieter mit einem räumlichen Verzeichnisdienst wider. Durch die Charakteristik der Anfragen können bei der Verarbeitung von Gebietsanfragen alle Datenanbieter gleichzeitig befragt werden. Bei der Verarbeitung von föderierten Nachbarschaftsanfragen kann der Ortsbezug sehr elegant zur Bestimmung des Anfragegebiets ausgenutzt werden. Die MRep-Relationenobjekte können mit der Position der zugehörigen Objekte versehen werden, und auf diese Weise sehr effizient mit Gebietsanfragen ermittelt werden. Die kooperierenden Föderationskomponenten nutzen den Ortsbezug als Partitionierungsmöglichkeit für die Zuständigkeitsbereiche. Zusätzlich führt der ähnliche Standort der mit der gleichen Basisstation verbundenen mobilen Nutzer zu einer Erhöhung der Trefferquote im Cache. Insgesamt wurde die Problemstellung der föderierten Anfrageverarbeitung in einer Integrationsmiddleware für orts- und kontextbezogene Anwendungen umfassend erörtert. Effiziente Verfahren, welche die Interaktion der Komponenten des Nexus-Systems beschreiben, wurden vorgestellt.

5

Realisierung der Anfrageverarbeitung

Nachdem im vorigen Kapitel die Interaktionen zwischen den einzelnen Komponenten im Nexus-System untersucht wurden, sollen in diesem Kapitel Konzepte und Verfahren zur Umsetzung der Anfrageverarbeitung innerhalb einer Föderationskomponente erörtert werden. Hierfür wird zunächst in Kapitel 5.1 eine geeignete Hauptspeicherrepräsentation für die zu verarbeitenden Daten vorgestellt. Diese stellt das Bindeglied zwischen allen weiteren in diesem Kapitel diskutierten Konzepten sowie zur föderierten Anfrageverarbeitung dar. In Kapitel 5.2 wird untersucht, wie die charakteristische Eigenschaft der typischen Anfragen in der Anwendungsdomäne der ortsbezogenen Anwendungen – die meisten Anfragen schränken unter anderem Ort und Typ der gesuchte Objekte ein – zur Steigerung der Effizienz der Anfrageverarbeitung im Hauptspeicher ausgenutzt werden kann. Die Daten verschiedener Anbieter können in verschiedenen Koordinatensystemen vorliegen. Damit sie im Hinblick auf Interoperabilität und Föderation trotzdem gemeinsam verarbeitet werden können, wird in Kapitel 5.3 beschrieben, wie eine automatische Transformation zwischen den verschiedenen Koordinatensystemen bewerkstelligt werden kann. In Kapitel 5.4 werden verschiedene Konzepte zur Integration domänenspezifischer Funktionalität in die Anfrageverarbeitung vorgestellt, sowie die sich bereits im Einsatz befindlichen konkreten Umsetzungen präsentiert. Diese integrieren zum einen die Daten verschiedener Anbieter und bieten zum anderen verschiedene Möglichkeiten zur Aufbereitung der Daten. Weitere Maßnahmen zur Steigerung der Effizienz der Anfrageverarbeitung innerhalb einer Föderationskomponente werden in Kapitel 5.5 diskutiert. Hierbei werden insbesondere verschiedene Konzepte zum Zwischenspeichern von Ergebnismengen erörtert. Eine Bewertung der vorgestellten Konzepte schließt dieses Kapitel ab.

5.1 Repräsentation der Objektdaten im Hauptspeicher

Zur Verarbeitung von Anfragen in der Föderationskomponente ist es unerlässlich, dass Anfragen und Umgebungsmodelldaten, welche die Föderationskomponente in XML-kodierten Datenaustauschformaten (siehe Kapitel 3.2 und 3.3) erreichen, auch im Hauptspeicher adäquat repräsentiert werden können. Alle in der Föderationskomponente ausgeführten Schritte der Anfrageverarbeitung, wie beispielsweise die Transformation einer Anfrage in die disjunktive Normalform oder die Verschmel-

zung der Ergebnismengen mehrerer Datenanbieter, arbeiten auf diesen Hauptspeicherrepräsentationen. Zu diesem Zweck wurden eine Reihe von Java Klassenbibliotheken erstellt, die im Folgenden vorgestellt werden. Die AWS Klassen repräsentieren das Schema des Umgebungsmodells, die AWML Klassen repräsentieren die Umgebungsmodelldaten und die AWQL Klassen repräsentieren Anfragen an das Umgebungsmodell. Die Klassen unterstützen dabei auch die in [HKNS05] vorgestellte Erweiterung des Datenmodells, siehe Kapitel 2.3.

5.1.1 AWS Klassen

Die AWS Klassen repräsentieren das Schema des Umgebungsmodells im Hauptspeicher. Dies wird benötigt, um zu einer gegebenen Anfrage überprüfen zu können, ob die genannten Attribute und Typen auch gültig sind, und ob die Basisdatentypen der Attributteile zum Vergleichsprädikat passen. Des Weiteren kann damit bei Objekten, die mittels der AWML Klassen repräsentiert werden, überprüft werden, ob alle geforderten Attribute enthalten sind, ob die restlichen Attribute zum Objekttyp passen, und ob alle Attributteile den richtigen Basisdatentyp haben und im gültigen Wertebereich liegen.

Entsprechend dem in Kapitel 2.3 vorgestellten Datenmodell und insbesondere in Anlehnung an die in [BDG+04] definierten XML Schemata zur Beschreibung des Schemas des Umgebungsmodells sind die AWS Klassen wie folgt gegliedert. Die Klasse BasicDataType ist eine Aufzählung der möglichen Basisdatentypen. Die Klasse AttributePartBasicType kombiniert einen BasicDataType mit einem semantisch aussagekräftigen Namen und optional mit Einschränkungen des Wertebereichs. Die Klasse AttributePartTypeDef ordnet einem Namen für einen Attributteil den zugehörigen AttributePartBasicType zu. Der AttributeBasicType stellt eine Schablone für ein Attribut dar und legt fest, aus welchen Attributteilen das Attribut besteht, und welche davon optional sind. Die Klasse AttributeTypeDef ordnet einem Namen für ein Attribut die zugehörige Schablone zu. Die Klasse ObjectTypeDef ordnet einem Typnamen eine Menge von verbindlichen und optionalen Attributen zu. Die Klasse AWS speichert für jede der zuvor genannten Klassen eine Liste, die den im Schema verwendeten Namen mit der entsprechenden Instanz der zugehörigen Klasse verknüpft.

Abbildung 45 zeigt das Klassendiagramm der AWS und AWML Klassen, in dem auch die Verknüpfungen zwischen diesen beiden Gruppen von Klassen dargestellt sind.

5.1.2 AWML Klassen

Die AWML Klassen können alle Daten des Umgebungsmodells (siehe Kapitel 2.3) im Hauptspeicher repräsentieren. Sie sind wie folgt gegliedert. Die Klasse ResultSet speichert eine Ergebnismenge und kann beliebig viele Objekte des Umgebungsmodells enthalten, die jeweils verschiedene Objekttypen haben dürfen und aus verschiedenen Attributen zusammengesetzt sein dürfen. Auf einer Ergebnismenge können Anfragen verarbeitet werden, um eine Teilmenge daraus zu extrahieren.

Die Klasse GenericObject speichert ein einzelnes Objekt des Umgebungsmodells. Sie besteht im Wesentlichen aus einer Menge von Attributinstanzen. Dies dürfen

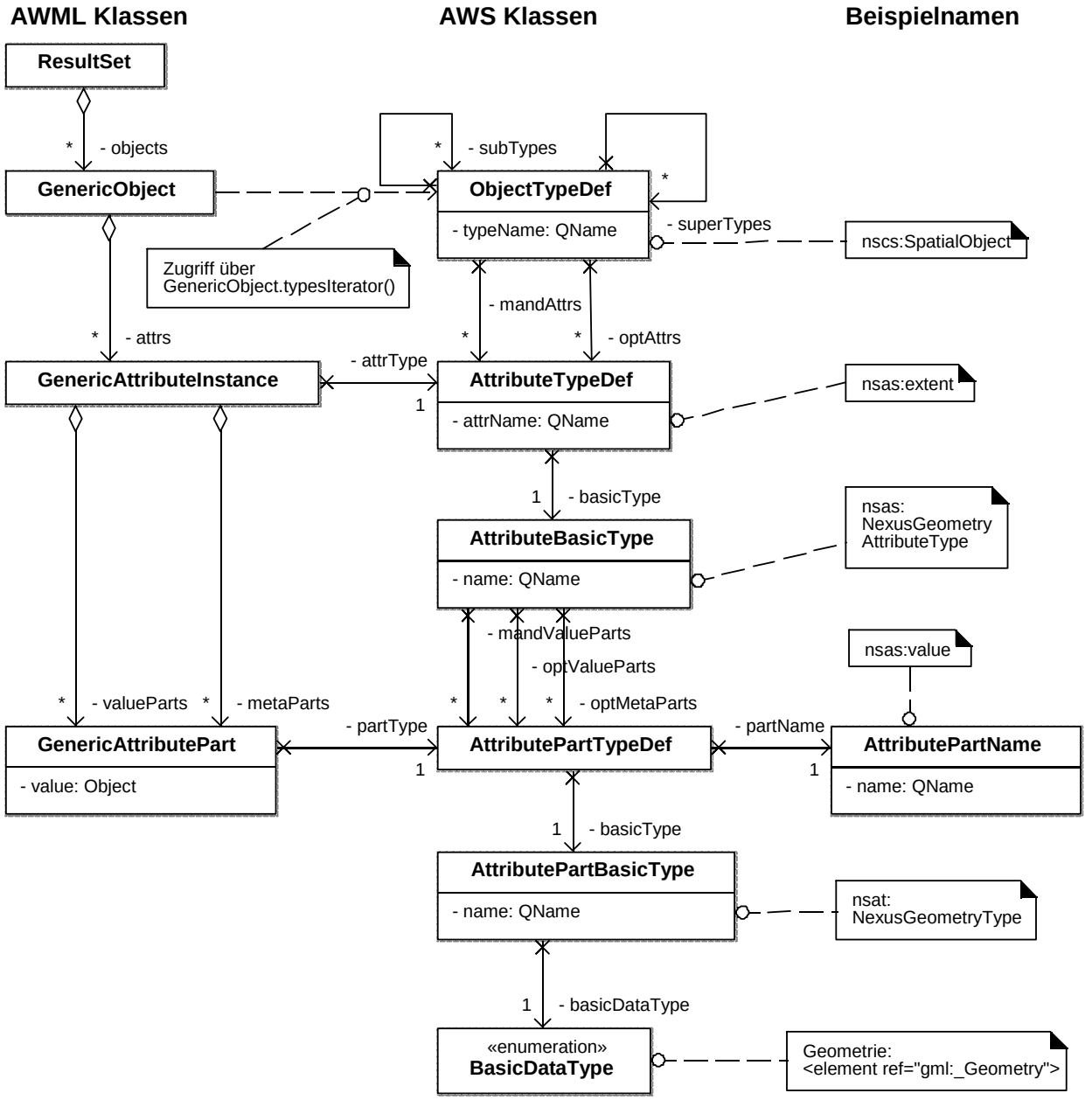


Abbildung 45: Klassendiagramm der AWML und der AWS Klassen

auch mehrere Instanzen des selben Attributs sein. Der Typ eines Objekts wird wie ein normales Attribut gespeichert. Der Basisdatentyp des zugehörigen Attributteils ist die Klasse ObjectTypeDef der AWS Klassen. Es existiert eine spezielle Komfortfunktion, mit der direkt auf den Typ eines Objekts zugegriffen werden kann. Eine weitere solche Funktion existiert für den Zugriff auf den Identifikator eines Objekts. Damit müssen nicht zuerst die entsprechenden Attributinstanzen und Attributteile gesucht werden.

Die Klasse GenericAttributeInstance verwaltet eine Instanz eines Attributs. Diese besteht aus ein oder mehreren Attributteilen. Der Attributtyp wird durch einen Verweis auf die AWS Klasse AttributeTypeDef festgelegt.

Die Klasse GenericAttributePart verwaltet einen Attributteil. Dieser besteht aus einem Verweis auf die AWS Klasse AttributePartTypeDef, wodurch der Name des

Attributteils und auch dessen Basisdatentyp festgelegt ist, und einem Verweis auf ein Java Objekt des Basisdatentyps, welches den Wert dieses Attributteils repräsentiert.

Auf diese Weise können Ergebnismengen, die aus Objekten bestehen, welche aus Attributinstanzen bestehen, welche wiederum aus Attributteilen bestehen (siehe Kapitel 2.3), im Hauptspeicher repräsentiert werden.

Interne Struktur

Der Aufbau der Klassen ist an das Java Collections Framework [Sun04] angelehnt. Die dort spezifizierten Methoden können genutzt werden, um die entsprechenden Java-Objekte hinzuzufügen und zu entfernen, und um über alle enthaltenen Java-Objekte zu iterieren. So kann bei einem Objekt des Umgebungsmodells über alle seine Attributinstanzen iteriert werden. Mit einer zusätzlichen Methode kann auch nur über die Instanzen eines vorgegebenen Attributs iteriert werden.

Die gleichen Möglichkeiten existieren bei einer Attributinstanz. Es kann entweder über alle enthaltenen Attributteile iteriert werden, oder es kann auf einen bestimmten Attributteil direkt zugegriffen werden. Die einzelnen Attributteile enthalten jeweils nur einen einzelnen Wert und stellen somit keine Java Collection dar.

Eine Ergebnismenge ist eine Sammlung von Objekten. Sie unterstützt ebenfalls die Java Collection Schnittstelle. Es gibt jedoch keine Komfortfunktionen, um gezielt auf einzelne Objekte zuzugreifen. Stattdessen können Anfragen auf der Ergebnismenge verarbeitet werden, um eine Teilmenge zu bilden, oder um bestimmte Objekte auszuwählen. Hierfür kann der *restriction* Teil einer AWQL Anfrage an die entsprechende Methode übergeben werden, oder es wird speziell für die Objektauswahl ein neuer *restriction* Teil erstellt. Bei der Verarbeitung einer solchen Anfrage kann die in Kapitel 5.2 vorgestellte orts- und typbezogene Anfrageverarbeitung genutzt werden. Zusätzlich sollen auch gewöhnliche Indexe auf anderen Attributen bzw. Attributteilen definiert werden können.

Intern werden Objekte in einer ArrayList verwaltet, um möglichst wenig Speicher zu belegen und um trotzdem den Komfort einer vollwertigen Liste nutzen zu können. Attributinstanzen und Attributteile werden in HashMaps verwaltet, um diese effizient mit deren Namen direkt ansprechen zu können.

Die Verbindung zur serialisierten XML Repräsentation (siehe Kapitel 3.2 und 3.3) schaffen, wie auch bei den anderen in diesem Teilkapitel vorgestellten Datenstrukturen, spezielle Parser- und Serialisiererklassen, welche dafür die StAX Technologie [Bea03] einsetzen.

Tatsächlich sind die AWML Klassen keine Klassen, sondern Schnittstellen. Die entsprechenden Klassen werden über das Factory Entwurfsmuster instanziiert, so dass die Implementierung transparent für den aufrufenden Programmteil ausgetauscht werden kann. Auf diese Weise können verschiedene interne Datenstrukturen und Verarbeitungsalgorithmen je nach Randbedingungen ausgewählt werden, ohne das nutzende Programm ändern zu müssen.

5.1.3 AWQL Klassen

Die AWQL Klassen können alle Teile einer AWQL-Anfrage im Hauptspeicher repräsentieren. Sie werden von der Föderation eingesetzt, um die Anfragen der Anwendungen intern zu repräsentieren und um sie umzuformen, beispielsweise in die disjunktive Normalform. Des Weiteren dienen sie als Eingabe für die Verarbeitung von Anfragen auf einer Ergebnismenge der AWML Klassen.

Den Kern der Klassen bildet eine Menge von Klassen, die alle von der Klasse RestrictionOperator abgeleitet sind. Diese repräsentieren einerseits die für die einzelnen Basisdatentypen wie Zeichenkette, Zahl, Geometrie oder Zeitpunkt vorgesehenen Vergleichsprädikate. Andererseits implementieren diese Klassen auch die Vergleichsverfahren und können überprüfen, ob der Wert in einem gegebenen Attributteil den Vergleich erfüllt.

Ein RestrictionOperator setzt sich aus einer Kennung der Vergleichsart, einem Pfad zum zu vergleichenden Wert sowie aus dem Vergleichswert selbst zusammen. Der Pfad besteht aus dem Attributnamen und dem Namen des Attributteils innerhalb dieses Attributs. Der Vergleichswert wird in einer Instanz der Klasse GenericAttributePart gekapselt.

5.1.4 Zusammenfassung

Die hier vorgestellten Klassen werden eingesetzt, um die Daten beliebiger Ergebnismengen zu speichern. In der Föderationskomponente sind dies die von den Datenanbietern gelieferten Ergebnismengen und die an die Anwendung geschickte Gesamtergebnismenge. Des Weiteren werden diese Klassen in der Schnittstelle der Ergebnisaufbereitungsoperatoren benutzt, um die zu verarbeitenden Objekte zu transportieren. Schließlich können auch die im Cache zwischengespeicherten Objekte hiermit repräsentiert werden. Außerhalb der Föderation können die Klassen von Mehrwertdiensten und Anwendungen zur Verarbeitung der Umgebungsmodelldaten genutzt werden. Ein hauptspeicherbasierter Datenanbieter auf Basis dieser Klassen existiert ebenfalls schon, und leistet gute Dienste beim Testen verschiedener Funktionalitäten innerhalb und außerhalb der Nexus-Plattform. Durch die enge Kopplung der AWML, AWQL und AWS Klassen fällt die Implementierung eines solchen Datenanbieters besonders leicht.

5.2 Orts- und typbezogene Anfrageverarbeitung

In den aufstrebenden Anwendungsgebieten der ortsbezogenen Anwendungen und des Ubiquitous Computing entstehen neue datenintensive Anwendungen, die den Benutzer gezielt durch die Bereitstellung der richtigen Informationen am richtigen Ort unterstützen. Sie liefern auf Abruf diejenigen Informationen, die gerade am besten auf die aktuelle Situation des Benutzers passen. Die Position des Benutzers und die Anwendung, die er gerade benutzt, bestimmen dabei, welche Informationen gerade relevant sind. Deshalb enthalten auch die meisten Informationsanfragen der Anwendungen räumliche Prädikate sowie Prädikate, die den Typ der gewünschten Daten festlegen. In diesem Kapitel wird deshalb eine spezielle orts- und typbezogene Anfrageverarbeitungskomponente eingeführt, die auf diese Randbedingungen

besonders zugeschnitten ist. Es wird ein Hauptspeicheransatz angestrebt, damit diese Komponente möglichst leicht installierbar ist und gegebenenfalls auch automatisch auf andere Knoten verlagert werden kann. Die folgenden Untersuchungen haben das Ziel, die für die interne Repräsentation der Daten in der Anfrageverarbeitungskomponente am besten geeignete Indexstruktur zu bestimmen [SGNM06].

Die hier untersuchte spezielle Anfrageverarbeitungskomponente kann auch in anderen bekannten Systemen eingesetzt werden. Implementierungen des OGC Catalogue Services Standards [OGC05a] oder eines FGDC clearinghouse [Nebe96], welche beide einen Verzeichnisdienst für digitale Geodaten bieten, können intern direkt die hier vorgestellte Anfrageverarbeitungskomponente einsetzen. Gleichermaßen können geographische Informationssysteme von so einer Komponente profitieren. Des Weiteren können die hier vorgestellten Ansätze auch auf die Anfrageverarbeitungskomponenten von Grid Metadata Catalog Services [Sing03] oder Verzeichnisdiensten in serviceorientierten Architekturen [Barr03] übertragen werden, welche jeweils Ressourcen oder Dienste nach gegebenen Bedingungen auswählen.

Um die Anfrageleistung von solchen verteilten Systemen zu erhöhen, ist es von Vorteil, wenn jede Daten verarbeitende Komponente, insbesondere auch die Datenquellen, selbst Anfragen verarbeiten kann, insbesondere Selektionsanfragen. Dies unterstreicht die Notwendigkeit, dass die Anfrageverarbeitung leicht installierbar sein muss. Weitere Vorteile bringt der Wechsel von einer generischen, für alle Eventualitäten vorbereiteten Anfrageverarbeitung, wie sie in typischen Datenbanksystemen zu finden ist, hin zu einer funktions- und anwendungsspezifischen Anfrageverarbeitung, wie sie hier vorgestellt wird. Die kann speziell auf die Eigenschaften typischer Anfragen zugeschnitten sein, und dabei ausnutzen, dass Selektionsanfragen für kontextbezogene Anwendungen ausreichend sind, siehe Kapitel 3.3.

5.2.1 Beitrag

In diesem Kapitel wird der Entwurf und die Implementierung einer hauptspeicherbasierten Anfrageverarbeitungskomponente beschrieben, welche intern eine Indexstruktur einsetzt, die sowohl die räumliche als auch die Typdimension nutzt, so dass orts- und typbezogene Anfragen bestmöglich unterstützt werden. Das Hauptaugenmerk liegt hierbei nicht auf der Entwicklung neuer Indexverfahren, sondern auf der Konfiguration bekannter Indexverfahren als interne Datenstrukturen, um die bestmögliche Indexorganisation zu erreichen. Deshalb werden vier verschiedene Ansätze, die interne Indexstruktur zu organisieren, untersucht, welche jeweils auf unterschiedliche Weise die räumliche Dimension und die Typdimension verbinden.

Viele Komponenten im Nexus-System können von der hier diskutierten Anfrageverarbeitungskomponente profitieren, vor allem auch wegen ihrer leichten Installierbarkeit. Für vier verschiedene Komponenten werden Einsatzszenarien mit unterschiedlichen Charakteristika entwickelt, und es wird jeweils der Gewinn durch die zugeschnittene Anfrageverarbeitung ermittelt. Damit eine Anwendung noch interaktiv benutzt werden kann, muss das Gesamtsystem Anfragen in weniger als einer Sekunde beantworten. Da darin auch Netzwerklatenzen, Serialisierung und Deserialisierung sowie weitere Verarbeitungsschritte enthalten sind hat eine

einzelne Anfrageverarbeitungskomponente nur etwa 10 Millisekunden Zeit, um eine Anfrage zu beantworten, die eine Ergebnismenge in der Größenordnung von 1000 Objekten zurückliefert. Dies alles spricht für eine Hauptspeicherlösung, um kurze Antwortzeiten und eine leichte Installierbarkeit zu erzielen.

Für jedes Einsatzszenario wurde eine erhebliche Menge an Messungen durchgeführt, um die jeweilige Eignung der verschiedenen Indexstrukturen zu ermitteln. Überraschenderweise hat sich dabei herausgestellt, dass die mehrdimensionale Indexstruktur, welche als am geeignetsten eingeschätzt wurde, tatsächlich um 10 bis 20 Prozent schlechter ist als ein anderer Ansatz, der in cleverer Weise konventionelle Indextechnologie einsetzt. Im Vergleich zu getrennten Indexen in der Typ- und Raumdimension kann die Leistung in bestimmten Fällen um bis zu eine Größenordnung gesteigert werden.

Im weiteren Verlauf des Kapitels werden zunächst in Kapitel 5.2.2 die zu verarbeitenden Daten und typische Anfragen eingeführt. Anschließend werden die Einsatzszenarien in Kapitel 5.2.3 charakterisiert. Die verschiedenen Ansätze zur Organisation der Indexstruktur werden in Kapitel 5.2.4 beschrieben. Die Messungen sowie deren Ergebnisse werden in Kapitel 5.2.5 diskutiert. In Kapitel 5.2.6 werden verwandte Arbeiten besprochen. Die Zusammenfassung in Kapitel 5.2.7 schließt dieses Themengebiet ab.

5.2.2 Daten und Anfragen

Typischerweise sind die Daten in den Anwendungsdomänen der orts- bzw. kontextbezogenen Anwendungen in Objekte strukturiert, siehe Abbildung 47. Bei geographischen Informationssystemen werden solche Objekte auch „features“ genannt. Das Schema der Daten besteht aus einer Menge von Typen. Ein Objekt ist mindestens einem Typ zugeordnet. In bestimmten Fällen, z. B. als Folge der Datenintegration, kann ein Objekt auch mehrere Typen haben. Ein Typ bestimmt die Namen und Datentypen der Attribute, in denen ein Objekt dieses Typs seine Daten speichert. Typen sind in einer Vererbungshierarchie angeordnet, wie beispielhaft in Abbildung 46 dargestellt. Wenn Typ B ein Subtyp von Typ A ist, dann erbt Typ B alle Attribute, die bei Typ A definiert sind. Zusätzlich kann Typ B noch weitere Attribute definieren. Typen dürfen auch von mehreren anderen Typen erben solange dadurch keine zyklische Vererbungsbeziehung entsteht. Weitere Details zum Datenmodell wurden in Kapitel 2.3 beschrieben.

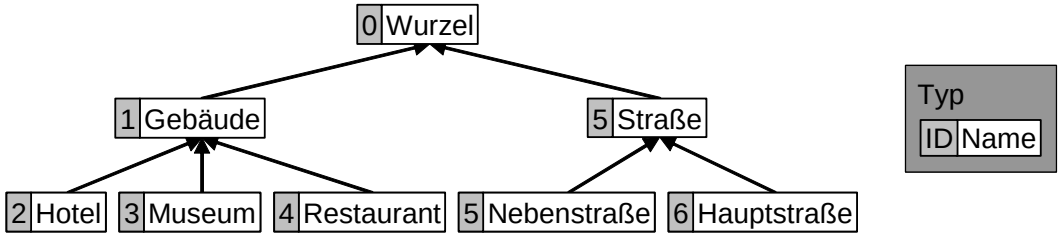


Abbildung 46: Vereinfachter Auszug aus einer typischen Typhierarchie

Jedem Typ wird eine eindeutige Identifikationsnummer zugeordnet, genannt Typ ID oder kurz ID. Durch eine optimale Zuordnung, welche in [MuPo97] Linearisierung genannt wird, kann für jeden Typ ein Intervall gefunden werden, das die ID

des Typen sowie die IDs aller seiner Subtypen enthält. Dies gelingt immer, solange in der Typhierarchie nur einfache Vererbungsbeziehungen existieren. In bestimmten Fällen kann auch bei Mehrfachvererbung eine Linearisierung gefunden werden [MuPo97]. Wenn die Typhierarchie zu komplex ist, so dass ein Typ nicht auf ein zusammenhängendes Intervall abgebildet werden kann, dann muss dieser Typ auf mehrere Intervalle abgebildet werden. Letztendlich ist dies vergleichbar mit einem Objekt, dem mehrere Typen zugeordnet sind.

Ohne Beschränkung der Allgemeinheit wird im Folgenden abstrahiert von Objekten, denen mehrere Typen zugeordnet sind, und von Anfragen, die mehrere Typen gleichzeitig anfordern. Spezielle Optimierungen hierfür werden nicht betrachtet. Ein Objekt mit mehreren Typen kann wie mehrere Objekte mit je einem Typ behandelt werden, indem es für jeden Typ separat in den Index aufgenommen wird. Die Anfrageverarbeitung muss dann um einen zusätzlichen Schritt ergänzt werden, welcher Duplikate in der Ergebnismenge eliminiert. Gleichermaßen können Anfragen nach mehreren Typen wie mehrere Anfragen nach je einem Typ behandelt werden. Die Implementierung kann solche Anfragen in einem einzigen Durchlauf bearbeiten. Zusätzlich kann in manchen Fällen auch der Duplikateliminierungsschritt entfallen, z. B. wenn der gesamte Datensatz sequentiell durchsucht wird.

Es wird davon ausgegangen, dass jedes Objekt eine Position oder eine Fläche hat, so dass bereits der Wurzeltyp der Typhierarchie ein entsprechendes geometrisches Attribut definiert, das zur Erstellung eines Indexes genutzt werden kann. Zahlreiche Beispiele untermauern diese Annahme wie das Datenmodell des TIGER/Line Datensatzes [UCB03], der kommende Standard für Stadtmodelle, CityGML [KGP05], oder das in Nexus verwendete Umgebungsmodell [NiMi04], siehe Kapitel 2.3. Die Geometrien der Objekte bestehen im Wesentlichen aus Linienzügen und Polygonen, welche für Indexierungszwecke alle durch minimale umschließende Rechtecke (engl. bounding boxes) angenähert werden können. Ein Datensatz kann z. B. verschiedene Arten von Straßen (Nebenstraße, Hauptstraße, Autobahn, ...), Gebäude, Sehenswürdigkeiten (Museum, Kirche, Aussichtspunkt, ...) usw. enthalten. Abbildung 47 zeigt einen Ausschnitt aus einem typischen Datensatz.

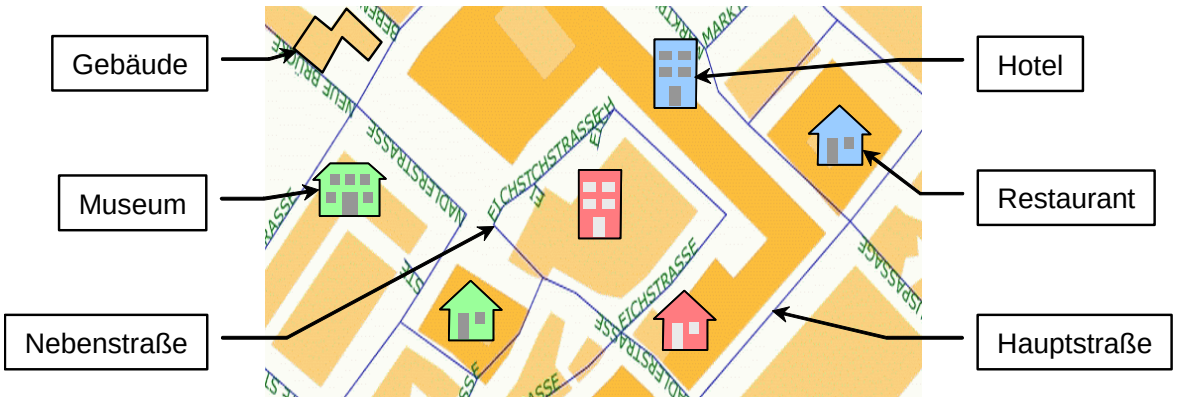


Abbildung 47: Ausschnitt aus einem typischen Datensatz

Typische Anfragen sind, ausgedrückt in natürlicher Sprache, „Liefere alle Straßenobjekte (egal welcher Art) in einem gegebenen Rechteck“, „Liefere alle Hauptstraßen in einem Streifen zwischen meiner aktuellen Position und meinem Ziel“ oder „Zeige alle Restaurants einer bestimmten Art im Umkreis von einem Kilometer“. Allen diesen Anfragen ist gemeinsam, dass sie ein räumliches Prädikat enthalten,

welches die Position der Ergebnisobjekte einschränkt, und dass sie ein Typprädikat enthalten, welches den Typ der Ergebnisobjekte festlegt. Die hier vorgeschlagenen Indexstrukturen sollen diese Gemeinsamkeiten der Anfragen ausnutzen. Typischerweise enthalten die Anfragen weitere Prädikate auf anderen Attributen. Hier ist jedoch mit wesentlich geringeren Überlappungen zwischen verschiedenen Anfragen zu rechnen, so dass es sich nicht lohnt, diese Attribute ebenfalls in eine kombinierte Indexstruktur aufzunehmen.

5.2.3 Einsatzszenarien

Im Nexus-System als Beispiel für ein großes kontextbezogenes Informationssystem kann die hier diskutierte Anfrageverarbeitungskomponente an vielen verschiedenen Stellen eingesetzt werden, siehe Abbildung 48.

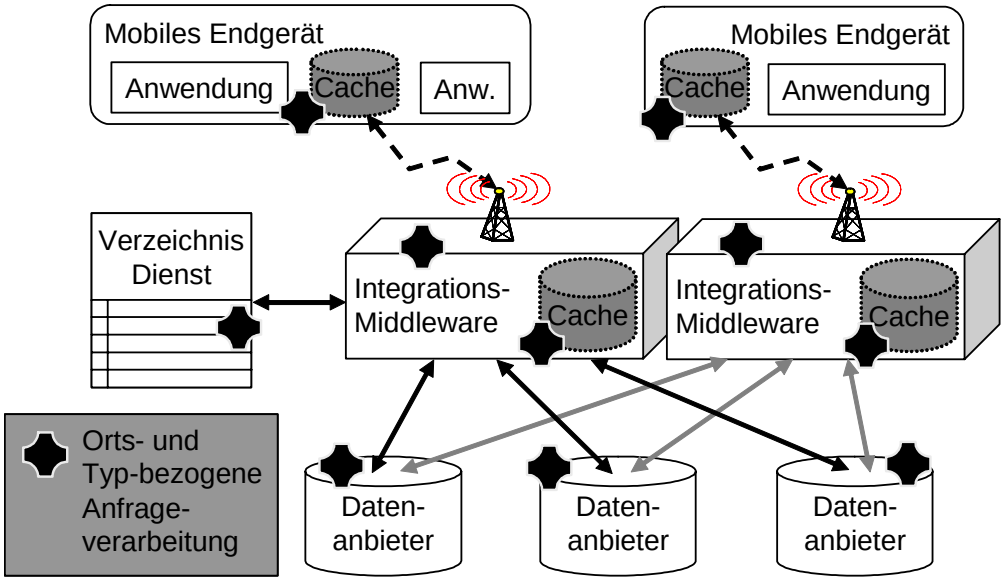


Abbildung 48: Einsatzorte der orts- und typbezogenen Anfrageverarbeitung im Nexus-System

Die einzelnen Einsatzorte werden im Folgenden entlang des Verarbeitungspfads einer typischen Anfrage vorgestellt. Eine Anwendung auf einem *mobilen Endgerät* stellt eine Anfrage nach Daten, die sich auf die nähere Umgebung des Benutzers beziehen. Diese Anfrage wird zuerst von der Anfrageverarbeitungskomponente des lokalen Caches bearbeitet. Eine Restanfrage nach den dann noch fehlenden Daten wird an die *Integrationsmiddleware* gestellt. Diese besitzt auch einen Cache, dessen Anfrageverarbeitung nun die Anfrage verarbeitet. Anschließend befragt die Integrationsmiddleware den *Verzeichnisdienst*, welcher selbst eine Anfrageverarbeitungskomponente betreibt, nach den relevanten Datenanbietern. Die Integrationsmiddleware leitet dann die an sie gestellte Anfrage an diese *Datenanbieter* weiter, welche die Anfrage auch mit einer orts- und typbezogenen Anfrageverarbeitungskomponente auswerten. Bei der Vereinigung der Ergebnismengen der verschiedenen Datenanbieter wird die Integrationsmiddleware wiederum von einer Anfrageverarbeitungskomponente unterstützt. Hierbei werden hauptsächlich Ergebnismengenaufbereitungsoperatoren und ortsbezogene Verschmelzungsalgorithmen unterstützt, siehe Kapitel 5.4.2 und Kapitel 4.4.1.2. Schließlich wird das integrierte Ergebnis an die Anwendung geschickt.

Wie schon zuvor erwähnt bestehen diese Anfrageverarbeitungskomponenten hauptsächlich aus einer geeigneten Indexstruktur, die prädikatbasierte Anfragen unterstützt, welche vor allem aus Orts- und Typprädikaten bestehen. Deshalb hängt die Leistung der Anfrageverarbeitungskomponente in erster Linie von der Leistung der intern eingesetzten Indexstruktur ab. Es ist leicht nachzuvollziehen, dass alle vier oben beschriebenen Komponenten (Mobiles Endgerät, Integrationsmiddleware, Verzeichnisdienst und Datenanbieter), welche jeweils eine Anfrageverarbeitungskomponente betreiben, von den hier untersuchten Indexstrukturen profitieren.

Jedoch verwaltet jede Komponente einen anderen Ausschnitt aus dem Gesamtdatenbestand. Auch unterscheiden sich die typischerweise gestellten Anfragen und die Häufigkeit der Änderungen im lokalen Datenbestand. Deshalb werden die Messungen in Kapitel 5.2.5 getrennt für jedes Einsatzszenario durchgeführt und jeweils individuell analysiert. Im Folgenden werden die Charakteristika der Einsatzszenarien beschrieben. Die Tabelle in Abbildung 49 quantifiziert diese Charakteristika, welche auf den Erfahrungen mit dem Nexus-System beruhen.

	Selektivitätsfaktor des räumlichen Prädikats	Selektivitätsfaktor des Typprädikats	Aktualisierungs-rate
Datenanbieter	1% - 20%	20% - 100%	0,01
Verzeichnisdienst	1% - 5%	1% - 20%	0,1
Integrationsmiddleware	10% - 50%	1% - 20%	10
Mobiles Endgerät	10% - 50%	10% - 100%	100

Abbildung 49: Minimale und maximale Selektivitätsfaktoren sowie Aktualisierungsraten der Einsatzszenarien

Der Begriff Selektivitätsfaktor (SF) bezieht sich auf das Verhältnis der Anzahl der Objekte in der Ergebnismenge zur Anzahl der Objekte im Datensatz (der Datensatzgröße). Wenn ein Prädikat einen niedrigen SF hat, dann erfüllen nur wenige Objekte die Anfrage. Ein hoher SF korrespondiert mit vielen qualifizierenden Objekten. Der Selektivitätsfaktor des räumlichen Prädikats bezieht sich auf das Verhältnis der Größe des Anfragegebiets zur Größe des Gesamtgebiets des Datenbestands, welches durch die konvexe Hülle um alle Geometrien der Objekte gegeben ist. Die Aktualisierungsrate beschreibt die Anzahl der Objekte, die im Mittel zwischen zwei aufeinanderfolgenden Anfragen aktualisiert werden. Beispielsweise bedeutet eine Aktualisierungsrate von 0,1, dass in einem Zeitraum, in dem zehn Anfragen verarbeitet werden, nur ein Objekt aktualisiert wird.

5.2.3.1 Datenanbieter

Es wird angenommen, dass ein Datenanbieter viele gleichartige Daten verwaltet, wie z. B. alle Straßen einer Stadt oder alle Filialen einer Restaurantkette in einem Land. Deshalb haben räumliche Prädikate in einer Anfrage typischerweise einen niedrigen SF und Typprädikate haben einen hohen SF. Es wird weiterhin angenommen, dass die Daten weitgehend statisch sind, dass sie sich also nur sehr selten ändern. Insbesondere ändern sich der Typ und die Position eines Objekts so gut wie gar nicht. Es wird deshalb angenommen, dass es nur eine Aktualisierung pro einhundert Anfragen gibt, was so gut wie vernachlässigbar ist.

5.2.3.2 Verzeichnisdienst

Der Verzeichnisdienst speichert Metadaten über alle angemeldeten Datenanbieter. Diese Metadaten bestehen aus einer Liste aller Typen der von einem Datenanbieter gespeicherten Objekte und einem Polygon, das die Geometrien aller dieser Objekte einschließt. Dieses Polygon wird auch Dienstgebiet genannt. Es wird angenommen, dass es viele Datenanbieter gibt, deren Dienstgebiete über das ganze Land verteilt sind, so dass die Selektivitätsfaktoren sowohl der räumlichen Prädikate als auch der Typprädikate typischerweise beide niedrig sind. Aktualisierungen treten hin und wieder auf, ungefähr einmal alle zehn Anfragen. Sie treten immer dann auf, wenn ein Datenanbieter seinen Dienst einstellt oder wenn sich ein neuer Datenanbieter beim System anmeldet.

5.2.3.3 Integrationsmiddleware

Die Integrationsmiddleware speichert selbst keine Daten. Sie unterhält jedoch einen Cache, in dem die Ergebnisse der verarbeiteten Anfragen zwischengespeichert werden. Es wird angenommen, dass die Integrationsmiddleware auf vielen Knoten des Gesamtsystems läuft, um eine große Anzahl an mobilen Endgeräten bedienen zu können. Es wird weiterhin angenommen, dass sich die Knoten, die die Integrationsmiddleware ausführen, die Arbeit in intelligenter Weise teilen. Beispielsweise könnte jeder Knoten einer Mobilfunkbasisstation zugeordnet sein, und nur diejenigen mobilen Endgeräte bedienen, die gerade mit dieser Basisstation verbunden sind. Da dadurch nur räumlich benachbarte Endgeräte bedient werden, wird angenommen, dass sich auch die angeforderten Daten zu einem großen Teil überlappen und somit eine hohe Trefferquote im Cache erreicht werden kann. Die Objekte im Cache haben deshalb viele verschiedene Typen, da auch viele verschiedene Anwendungen auf den Endgeräten laufen. Die Positionen der Objekte sind dagegen sehr dicht beieinander. Dementsprechend haben räumliche Prädikate einen hohen Selektivitätsfaktor, und Typprädikate einen niedrigen. Durch die räumliche Nähe der Objekte werden schätzungsweise im Mittel nur zehn Objekte pro Anfrage im Cache ausgetauscht.

5.2.3.4 Mobiles Endgerät

Trotz der begrenzten Ressourcen, die auf einem mobilen Endgerät verfügbar sind, betreibt es einen lokalen Cache, um mit weniger Mobilkommunikation auskommen zu können. Es wird angenommen, dass die im Cache gespeicherten Objekte alle ähnliche Typen haben, da typischerweise nur eine Anwendung pro Endgerät gleichzeitig läuft, die Anfragen nach immer den gleichen Objekttypen stellt. Dementsprechend sind die Selektivitätsfaktoren der räumlichen Prädikate und der Typprädikate beide hoch. Die Aktualisierungsrate ist höher als bei den Caches in der Integrationsmiddleware, da der Benutzer, der das mobile Endgerät mit sich führt, herumläuft und sich damit auch das Gebiet der aktuell relevanten Informationen ändert. Es werden schätzungsweise pro Anfrage einhundert Objekte im Cache ausgetauscht.

5.2.3.5 Zusammenfassung der Anforderungen

Aus der bisherigen Diskussion können die folgenden Anforderungen an die orts- und typbezogene Anfrageverarbeitung abgeleitet werden:

- Einfache Anfrageverarbeitungsfähigkeiten genügen. Anwendungen können leicht mit prädikatbasierten Selektionsanfragen auskommen.
- Typ und Geometrie kombinieren. Typische Anfragen enthalten räumliche Prädikate und Typprädikate.
- Verschiedene Arbeitslasten bewältigen. Die Selektivitätsfaktoren der Prädikate in typischen Anfragen und die Aktualisierungsrate hängen vom Einsatzszenario ab.
- Einfache Installierbarkeit. Viele Komponenten in einem großen Informationssystem können von der hier vorgestellten Anfrageverarbeitungskomponente profitieren.
- Kurze Antwortzeiten. Damit die Anwendungen trotz der komplexen Anfrageverarbeitung im Gesamtsystem noch interaktiv benutzt werden können, muss jede einzelne Anfrageverarbeitungskomponente Anfragen in der Größenordnung von zehn Millisekunden beantworten.

Die folgende Analogie verdeutlicht die Anfrageverarbeitungsfähigkeiten der orts- und typbezogenen Anfrageverarbeitungskomponente. Diese verhält sich im Vergleich zu einem kompletten Datenbanksystem wie eine XPath Verarbeitungskomponente zu einer XQuery Verarbeitungskomponente. Entsprechend wird die Leistung der Verarbeitungskomponente durch die Leistung ihrer internen Datenstrukturen bestimmt. Es wird ein Hauptspeicheransatz verfolgt, um kurze Antwortzeiten sowie eine leichte Installierbarkeit zu erzielen. Deshalb wird im Folgenden untersucht, wie die Hauptspeicherdatenstrukturen organisiert sein müssen, um die beste Leistung zu erreichen.

5.2.4 Indexstrukturen

In diesem Kapitel werden die verschiedenen Ansätze vorgestellt, die untersucht wurden, um eine Indexstruktur aufzubauen, welche die räumliche Dimension und die Typdimension kombiniert. Diese Kombination resultiert aus der Beobachtung in Kapitel 5.2.2, dass die Mehrheit der Anfragen Selektionsprädikate auf mindestens diese beiden Dimensionen enthält. Bei jedem Ansatz wird Schritt für Schritt erläutert, wie die Verarbeitung einer Anfrage abläuft.

In den folgenden Erläuterungen wird die räumliche Dimension als eine einzige Dimension betrachtet, obwohl sie tatsächlich aus zweidimensionalen Koordinaten besteht. Es wird auch von den Details einer bestimmten räumlichen Indexstruktur abstrahiert (wie z. B. R*-Baum, Grid-File oder MX-CIF Quadtree, siehe [GaGü98] für einen Überblick), da die eingesetzte räumliche Indexstruktur leicht gegen eine andere ausgetauscht werden kann, ohne die Leistung der Ansätze im Verhältnis zueinander nennenswert zu beeinflussen. Das Hauptaugenmerk liegt auf der neuartigen Kombination bekannter Indexstrukturen, um die vorliegende Aufgabe am besten zu bewältigen.

Die im Folgenden beschriebenen Ansätze sind alle hauptspeicherbasiert wegen der Anforderung nach sehr kurzen Antwortzeiten. Durch Partitionieren des Gesamtdatenbestands kann dieser so aufgeteilt werden, dass jede Komponente ihre Daten im Hauptspeicher verwalten kann. Des Weiteren erlaubt dieser hauptspeicherbasierte Ansatz die flexible Installation der Anfrageverarbeitungskomponente in jede Komponente des Gesamtsystems mit wesentlich geringerem Verwaltungsaufwand als bei der Installation eines kompletten Datenbanksystems notwendig wäre. Neben der Anfrageverarbeitungskomponente müssen keine weiteren Komponenten wie beispielsweise ein Datenbanksystem installiert und konfiguriert werden.

Alle Ansätze verwenden Hilfsdatenstrukturen, welche den Namen eines Typs auf dessen Identifikationsnummer (ID) abbilden, sowie eine ID auf je eine Liste aller zugehörigen Sub- bzw. Supertypen abbilden. Durch den Einsatz von Hash-Datenstrukturen erfolgen diese Abbildungen jeweils in konstanter Zeit ($O(1)$). Die Größe und der Inhalt dieser Hilfsdatenstrukturen hängt nur von der Typhierarchie ab, so dass diese Datenstrukturen sehr klein sind im Vergleich zu den zu verwaltenden Daten. Aktualisierungen des Datenbestands haben ebenfalls keine Auswirkungen auf diese Datenstrukturen.

5.2.4.1 Getrennte Indexe (GI)

Der „Getrennte Indexe“ Ansatz unterhält zwei getrennte Datenstrukturen, wie in Abbildung 50 dargestellt. Die erste Datenstruktur ordnet mittels eines räumlichen Indexes die Objekte lediglich nach ihrer Geometrie. Die zweite Datenstruktur besteht aus einem Array, welches für jeden Typ die zugehörigen Objekte in einer Liste speichert. Neue Objekte müssen immer gleichzeitig in beide Datenstrukturen eingefügt werden, damit diese konsistent bleiben. Wenn ein Objekt mehrere Typen besitzt, dann wird das Objekt in jede der den Typen entsprechenden Listen in der zweiten Datenstruktur eingetragen. In den räumlichen Index wird das Objekt trotzdem nur einmal eingetragen.

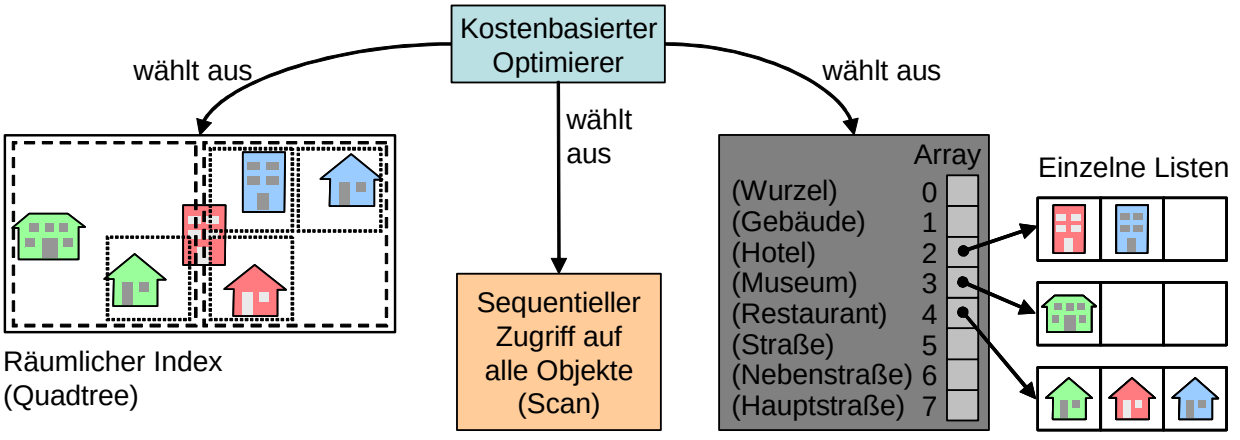


Abbildung 50: Datenstruktur des „Getrennte Indexe“ Ansatzes

Bei der Verarbeitung einer Anfrage bewertet ein kostenbasierter Optimierer (siehe u.a. [HäRa01]) die Selektivität des räumlichen Prädikats und des Typprädikats. Das selektivere Prädikat (das mit dem niedrigeren Selektivitätsfaktor) wird ausgewählt, und mit Hilfe der zugehörigen Datenstruktur wird eine Liste von Kandidaten

erstellt. Anschließend werden diese Kandidaten einer nach dem anderen mit dem verbleibenden Prädikat gefiltert. Übrig bleibt die endgültige Ergebnismenge. Alternativ kann der kostenbasierte Optimierer beschließen, keinen der Indexe zu verwenden sondern sequentiell jedes Objekt einzeln zu betrachten und jeweils mit beiden Prädikaten zu filtern. Dies ist sinnvoll, wenn beide Prädikate einen hohen Selektivitätsfaktor aufweisen und die Ergebnismenge annähernd den gesamten Datenbestand enthält.

Dieser Ansatz setzt nur gewöhnliche Datenbanktechnologie ein ohne irgendwelche problemspezifischen Verbesserungen. Dies ist damit der Referenzansatz, dessen Leistung alle anderen Ansätze mindestens erreichen sollten. Meist ist dieser Ansatz recht langsam, da er nur für eine Dimension eine Indexstruktur ausnutzt und alle Kandidaten entlang der anderen Dimension einzeln filtert.

In den Messungen wurde ein idealer Optimierer implementiert, indem für jede Messung alle drei Entscheidungsmöglichkeiten des Optimierers ausgeführt wurden und jeweils die kürzeste Zeit als Messwert weiterverwendet wurde. So können Fehlprognosen des Optimierers ausgeschlossen werden, und es fließt immer die bestmögliche Zeit in den Vergleich mit den anderen Ansätzen ein.

5.2.4.2 Räumlicher Index pro Typ (RIpT)

Der „Räumlicher Index pro Typ“ Ansatz nützt aus, dass die Typen der Objekte (bzw. deren Identifikationsnummern) diskret sind und keine Werte in einem kontinuierlichen Spektrum. Des Weiteren ist typischerweise die Anzahl an Typen überschaubar und bewegt sich in einer Größenordnung von unter 1000. Deshalb unterhält dieser Ansatz einen eigenen räumlichen Index für jeden Typ in der Typhierarchie, siehe Abbildung 51. Jedes Objekt wird in den räumlichen Index eingefügt, der mit seinem Typ assoziiert ist. Hat ein Objekt mehrere Typen, so wird es in jeden der zugehörigen räumlichen Indexe eingefügt. Auf diese Weise entsteht ein Wald von räumlichen Indexen, in dem jeder einzelne Index verhältnismäßig klein ist (je nach Verteilung der Objekte auf die Typen).

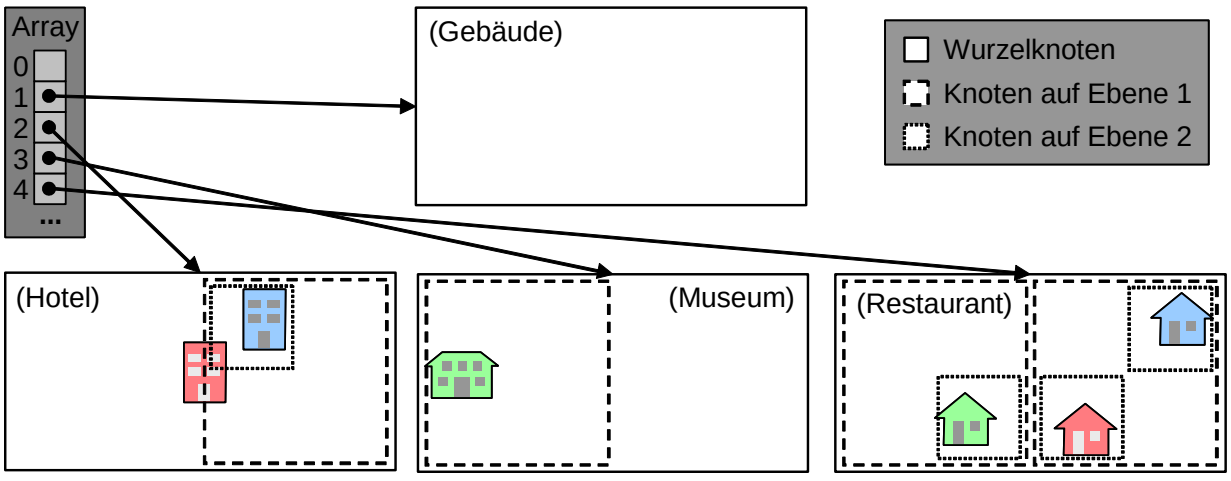


Abbildung 51: Datenstruktur des „Räumlicher Index pro Typ“ Ansatzes

Bei der Beantwortung einer Anfrage wird zunächst der angeforderte Typ in eine Liste umgewandelt, welche die Identifikationsnummern dieses Typs und aller sei-

ner (auch transitiven) Subtypen enthält. Danach wird auf jeden der zugehörigen räumlichen Indexe zugegriffen und jeweils eine Liste mit Kandidaten erstellt, die das räumliche Prädikat der Anfrage erfüllen. Schließlich werden alle Kandidaten in eine einzige Ergebnismenge zusammengeführt und doppelte Kandidaten eliminiert. Duplikate können jedoch nur auftreten, wenn Objekte mehrere Typen haben oder wenn Mehrfachvererbung in der Typhierarchie vorhanden ist.

Verglichen mit den „Getrennte Indexe“ Ansatz ist der Aufwand zur Verwaltung vieler kleiner räumlicher Indexe relativ klein. Zusätzlich erlaubt diese Organisation eine viel präzisere indexgestützte Auswahl der relevanten Objekte. Einige objektrelationale Datenbanksysteme wie z. B. PostgreSQL organisieren Indexe auf Hierarchien von typisierten Tabellen tatsächlich in dieser Weise. Die Leistung dieses Ansatzes wird schwächer, wenn die Typhierarchie umfangreich und tief ist, und wenn Anfragen überwiegend die „oberen“ Typen in der Hierarchie auswählen, so dass jeweils sehr viele räumliche Indexe traversiert werden müssen.

5.2.4.3 Aggregierter räumlicher Index pro Typ (ArIpT)

Der „Aggregierter Räumlicher Index pro Typ“ Ansatz zielt darauf ab, den Nachteil des vorigen Ansatzes auszumerzen, welcher auf viele einzelne räumliche Indexe zugreifen muss. Hierzu wird für jeden Typ der Typhierarchie ein eigener räumlicher Index angelegt, welcher jeweils alle Objekte enthält, die diesen Typ oder einen seiner Subtypen haben, siehe Abbildung 52. Jedes Objekt wird also in mehrere räumliche Indexe eingefügt: in den zum Typ des Objekts zugehörigen Index und in alle Indexe, die zu Supertypen dieses Typs gehören. Dies impliziert, dass im Index, der zum Wurzeltyp der Typhierarchie gehört, alle Objekte des Datenbestands enthalten sind.

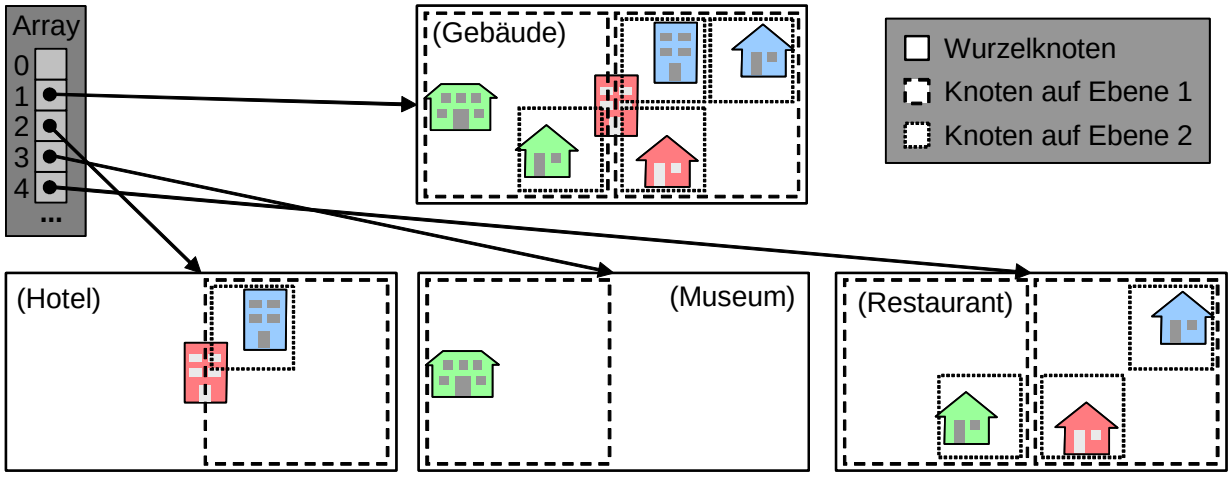


Abbildung 52: Datenstruktur des „Aggregierter Räumlicher Index pro Typ“ Ansatzes

Diese umfassende Redundanz macht sich bezahlt bei der Verarbeitung von Anfragen. Bei Anfragen nach einem einzelnen Typen kann die Anfrageverarbeitungskomponente immer einen räumlichen Index finden, der genau die passenden Objekte enthält: Objekte, die den geforderten Typ oder einen Subtyp davon aufweisen. Wenn in der Anfrage mehrere verschiedene Typen angefordert werden, dann muss für jeden Typ der entsprechende Index besucht werden.

Die Vorteile bei der Anfrageverarbeitung müssen durch Nachteile hinsichtlich des Speicherverbrauchs und des Verwaltungsaufwands erkauft werden. Die Höhe der Typhierarchie bestimmt den Grad der Redundanz, und damit den Aufwand, der für das Einfügen oder Löschen eines Objekts anfällt. In den Messungen zeigt sich, dass die Kosten für die Verwaltung von Aktualisierungen schon bei relativ kleinen Aktualisierungsraten die Leistungsgewinne bei Anfragen übersteigen.

Des Weiteren können Anfragen nach exakt dem geforderten Typ (ohne dessen Subtypen) nur in sehr ineffizienter Weise verarbeitet werden. Die Objekte, die nur einen Subtyp des geforderten Typen haben, müssen mühsam in einem zusätzlichen Filterschritt wieder von der initialen Ergebnismenge entfernt werden.

5.2.4.4 Dreidimensionaler Index (3DI)

Der „Dreidimensionaler Index“ Ansatz hat das Ziel, die Vorteile der beiden zuvor diskutierten Ansätze zu kombinieren, ohne deren Nachteile zu erben. Wie in Abbildung 53 gezeigt wird, verwendet dieser Ansatz eine einzige räumliche Indexstruktur mit drei Dimensionen. Die ersten beiden Dimensionen repräsentieren die räumlichen Dimensionen der Geometrien der Objekte in gleicher Weise wie die räumlichen Indexe der anderen Ansätze. Die dritte Dimension repräsentiert die Identifikationsnummern der Typen, bzw. deren Abbildung in den Wertebereich der Typdimension. Ein Objekt wird in diesen Index als flache Scheibe eingefügt, wobei die Ausdehnung der Scheibe der Geometrie des Objekts entspricht, und die Position der Scheibe in der dritten Dimension durch die Abbildung der Identifikationsnummer des Objekttyps in den Wertebereich der Typdimension festgelegt wird. Besitzt ein Objekt mehrere Typen so werden entsprechend viele Scheiben eingefügt.

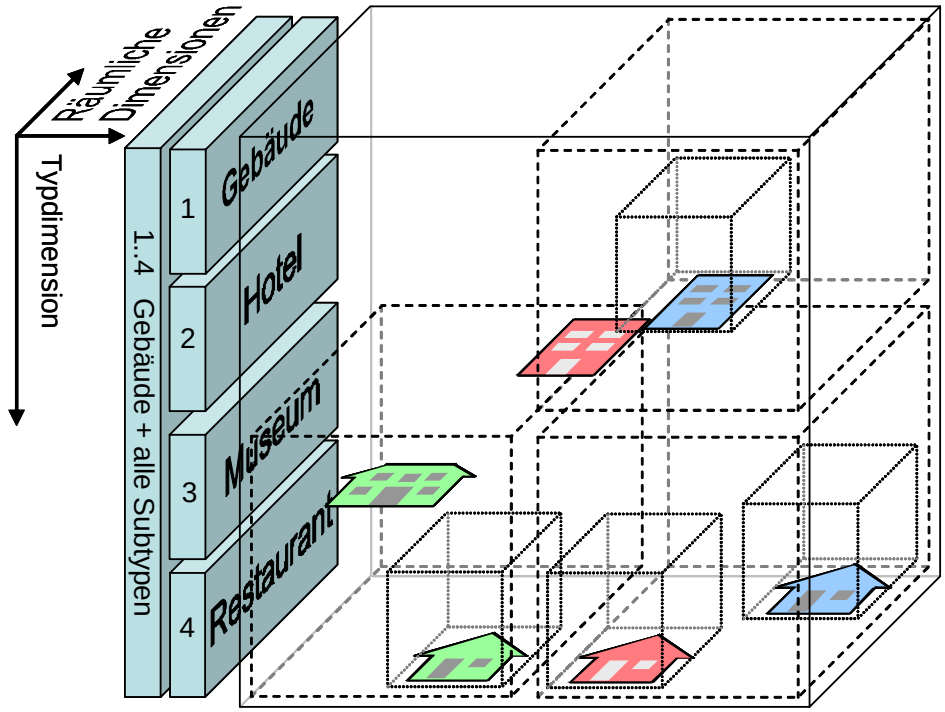


Abbildung 53: Datenstruktur des „Dreidimensionaler Index“ Ansatzes

In diesem Ansatz kann eine Anfrage, die aus einem räumlichen Prädikat und einem Typprädikat besteht als Quader im dreidimensionalen Raum dargestellt werden. Die Grundfläche des Quaders entspricht dem minimalen umschließenden Rechteck (engl. bounding box) der Anfragegeometrie. In der dritten Dimension nimmt der Quader einen Bereich ein, welcher den abgebildeten Werten des gesuchten Typs und dessen Subtypen entspricht. Wenn die Typhierarchie einfach genug ist (vergleiche Kapitel 5.2.2) kann sie optimal linearisiert werden, so dass jede Anfrage als ein einziger Quader repräsentiert werden kann, und die Anfrage mit einer einzigen Traversierung des dreidimensionalen Indexes beantwortet werden kann. Bei einer komplexeren Typhierarchie müssen mehrere Anfragequader erstellt werden und die Indexstruktur muss entsprechend oft traversiert werden. Als Optimierung kann der Anfragequader auch mehrere Intervalle in der Typdimension zulassen, so dass wiederum eine einzige Traversierung der Indexstruktur genügt, um eine Anfrage zu beantworten.

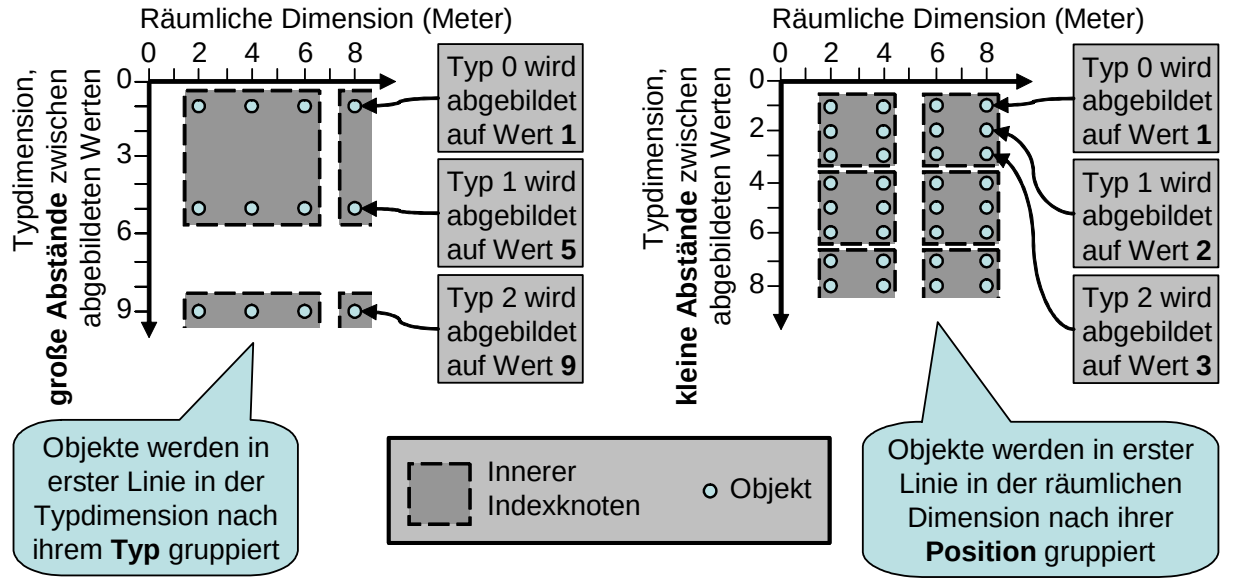


Abbildung 54: Auswirkungen verschieden großer Abstände zwischen den abgebildeten Werten in der Typdimension auf die Gruppierung der Objekte in den inneren Indexknoten

Die Abbildung der Typen bzw. deren Identifikationsnummern in den Wertebereich der Typdimension ist der springende Punkt bei diesem Ansatz. Abbildung 54 veranschaulicht dies. Der Abstand zwischen den abgebildeten Werten zweier Typen beeinflusst die Gruppierung von Objekten und untergeordneten Knoten in den inneren Knoten der Indexstruktur. Wenn der Abstand zwischen den abgebildeten Werten zweier Typen groß ist (linke Seite von Abbildung 54), dann werden die Objekte primär nach ihrem Typ gruppiert und nur nachrangig nach ihrer Position. Im Beispiel landen Objekte mit drei verschiedenen Positionen und mit nur zwei verschiedenen Typen im selben inneren Indexknoten. Bei der Verarbeitung von Anfragen können so innere Indexknoten, die den falschen Bereich in der Typdimension abdecken, schnell übersprungen werden. Wenn die abgebildeten Werte zweier Typen nahe beieinander liegen (kleine Abstände, rechte Seite von Abbildung 54), dann verhält es sich umgekehrt. Im Beispiel enthält jetzt ein innerer Indexknoten Objekte mit nur zwei verschiedenen Positionen aber drei verschiedenen Typen. Dieses Verhalten liegt darin begründet, dass die meisten Indexverfahren versuchen, die minimalen umgebenden Rechtecke der inneren Indexknoten so quadratisch wie

möglich zu halten. Wie die Experimente in Kapitel 5.2.5.2 zeigen werden, haben die Abstände zwischen den abgebildeten Werten der Typen einen großen Einfluss auf die Leistung der Indexstruktur. Bei großen Abständen können Teilbäume, welche die falschen Typen abdecken, früh übersprungen werden. Bei kleinen Abständen gilt das Gleiche für Teilbäume, die ein zu weit entferntes Gebiet abdecken.

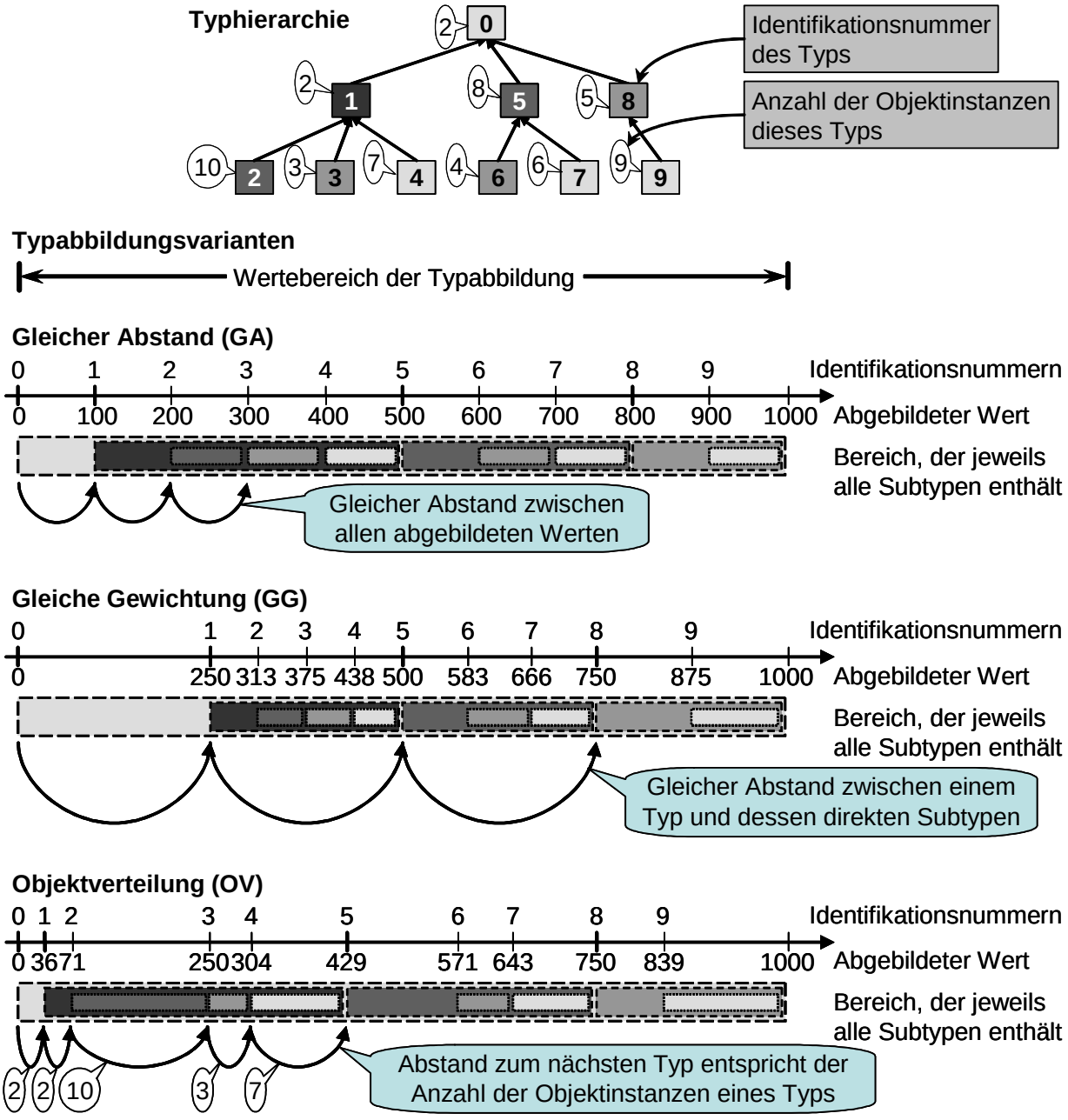


Abbildung 55: Berechnung der abgebildeten Werte in der Typdimension für die verschiedenen Typabbildungsvarianten

Es wurden drei verschiedene Abbildungsvarianten untersucht, welche die Abstände zwischen den abgebildeten Werten der Typen auf unterschiedliche Weise berechnen, siehe Abbildung 55. In allen drei Varianten werden die abgebildeten Werte so skaliert, dass sie den vorgegebenen Wertebereich der Typabbildung ganz ausfüllen. Dieser Wertebereich wird in Kapitel 5.2.5.2 näher erläutert. Die gestri-

chelten Kästen in Abbildung 55 deuten für jeden Typ den Bereich an, in dem die abgebildeten Werte seiner Subtypen liegen.

Die Varianten unterscheiden sich wie folgt:

- **Gleicher Abstand** für jeden Typ (GA): Der abgebildete Wert eines Typs entspricht seiner Identifikationsnummer, die um einen konstanten Faktor skaliert wurde. Dies ist die einfachste Variante, welche die Hierarchie der Objekttypen weitgehend unberücksichtigt lässt. Diese Variante ist sehr ähnlich zu dem Ansatz aus [MuPo97], bei dem jedoch die abgebildeten Werte nicht skaliert werden. Im Beispiel in Abbildung 55 gibt es zehn Typen und der Wertebereich der Typabbildung ist 1000 Einheiten groß, so dass der Abstand zwischen zwei abgebildeten Werten 100 Einheiten beträgt.
- **Gleiche Gewichtung** auf jeder Ebene der Typhierarchie (GG): Bei jedem Typ der Typhierarchie wird das diesem Typ von dessen Supertyp zugeordnete Intervall gleichmäßig auf diesen Typ und dessen direkte Subtypen aufgeteilt. In Abbildung 55 hat der Typ 0 drei direkte Subtypen (1, 5 und 8), so dass der zur Verfügung stehende Wertebereich der Typabbildung von 1000 Einheiten in vier gleich große Abschnitte aufgeteilt wird. Bei Typ 5 wiederum wird der zur Verfügung stehende Wertebereich (500 bis 749) in drei Abschnitte aufgeteilt, weil dieser Typ zwei Subtypen besitzt. Damit soll erreicht werden, dass bevorzugt Objekte mit dem gleichen Supertyp gruppiert werden.
- Gewichtung entsprechend der **Objektverteilung** (OV): Bei jedem Typ der Typhierarchie wird das diesem Typ von dessen Supertyp zugeordnete Intervall entsprechend der Anteile der tatsächlich jeweils vorhandenen Objektinstanzen auf diesen Typ und dessen direkte Subtypen aufgeteilt. In Abbildung 55 gibt es insgesamt 56 Objektinstanzen. Zwei Objektinstanzen haben Typ 0, so dass dessen Abstand zu Typ 1 gerundet $\frac{2}{56} \cdot 1000 \approx 36$ Einheiten beträgt. Damit diese Variante funktioniert, müssen entweder die Objekte gezählt werden oder entsprechende Statistiken erhoben werden. Damit kann die Gruppierung der Objekte in inneren Knoten entsprechend der tatsächlichen Objektverteilung beeinflusst werden. Häufige Typen bekommen schon sehr weit oben im Indexbaum einen eigenen Zweig während seltene Typen lange nur nach der Geometrie gruppiert werden und die Aufspaltung nach Typen erst sehr weit unten im Baum stattfindet, siehe hierzu auch Abbildung 54.

5.2.5 Messungen

Um die Leistung der verschiedenen Indexstrukturen bewerten zu können, wurden alle in Java implementiert. Als räumliche Indexstruktur wurde zunächst die R*-Baum-Implementierung der XXL Bibliothek [VBD+01] eingesetzt. Diese Implementierung verarbeitet Anfragen sehr schnell, benötigt aber für das Einfügen oder Aktualisieren von Objekten viel Zeit, da jeweils der Baum neu balanciert wird. Selbst wenn der gesamte Datenbestand vorsortiert wird und dann am Stück der Index aufgebaut wird (engl. sort-based bulk loading) ist der Aufwand noch erheblich. Deshalb wurde der R*-Baum gegen einen MX-CIF Quadtree [Same90] ersetzt, welcher mit sehr geringem Aufwand neue Objekte einfügen kann. Die Nachteile des Quadtrees liegen bei der Verwaltung von extrem ungleichmäßigen und entarteten

Datensätzen. Bei den hier untersuchten realen Datensätzen kamen sie nicht zum Tragen. Es wurde die Quadtree-Implementierung der JTS Topology Suite [ViSo04] adaptiert, so dass sie mehr als zwei Dimensionen unterstützt. Des Weiteren wurden einige Optimierungen vorgenommen, welche die Anfrageleistung auf das drei bis fünf-fache steigern konnten, so dass schließlich die Quadtree Implementierung sowohl beim Einfügen als auch beim Abfragen von Objekten schneller war als die R*-Baum-Implementierung. Da die tatsächlich eingesetzte räumliche Indexstruktur nur einen geringen Einfluss auf die relativen Leistungen der untersuchten Ansätze hat, werden im Folgenden nur die Ergebnisse der Messungen mit dem MX-CIF Quadtree diskutiert. Zusätzlich ist diese Implementierung auch wesentlich schneller als die R*-Baum-Implementierung.

Die Messungen wurden auf einem Doppelprozessor-Server von Dell durchgeführt, welcher zwei Intel Xeon Prozessoren mit 2GHz und 2GB Speicher enthält. Die Hälfte des Speichers wurde der Java Laufzeitumgebung zur Verfügung gestellt. Alle Messungen haben nur einen der beiden Prozessoren genutzt, so dass der andere Prozessor für die restlichen parallel laufenden Programme und das Betriebssystem zur Verfügung stand, wodurch die Messungenauigkeiten auf ein Minimum reduziert werden konnten. Um die benötigte Präzision zur Messung von Antwortzeiten im Bereich von unter einer Millisekunde zu erreichen, wurde das High Resolution Timer Paket [Roub03] eingesetzt.

Die Messungen wurden auf den TIGER/Line Datensätzen der amerikanischen Behörde für Bevölkerungsstatistik (US Census Bureau, [UCB03]) durchgeführt. Es wurden die Datensätze von neun verschiedenen Regierungsbezirken in Kalifornien ausgewählt. Abbildung 56 zeigt deren Charakteristika. Aus diesen Datensätzen wurden alle Objekte extrahiert, die einen Linienzug oder ein Polygon als Geometrie aufwiesen, so dass die untersuchten Datensätze zwischen 12000 und 200000 Objekte enthalten.

			Universum: Ausdehnung des Datensatzes (in km)		Anzahl der
Abkürzung	**Regierungsbezirk**	**Größe**	**Breite**	**Länge**	**Objekte**
#1	Yuba	klein	33,2	27,6	11923
#2	Glenn		42,6	70,8	16839
#3	San Francisco		15	78,4	22666
#4	Alameda	mittel	37,6	49,7	46285
#5	Santa Clara		42,1	42,8	53727
#6	Sacramento		49,2	45,5	71743
#7	Riverside	groß	59	26,5	151489
#8	Kern		98,6	17	175082
#9	San Diego		90,6	114,8	203122

Abbildung 56: In den Messungen verwendete Datensätze aus dem Bestand der TIGER/Line Datensätze

Die Datensätze enthalten unter anderem Daten über Straßen, Schienenstrecken, andere Beförderungsmittel am Boden, Landmarken, das Gewässernetz und Grundstücksgrenzen. Jedes Objekt besitzt einen Typ, welcher dort Census Class Feature Code (CFCC, [UCB02]) genannt wird. Diese Typen untergliedern sich in Haupt-

gruppen und Untergruppen. Diese Gruppierung wurde als eine Typhierarchie interpretiert, welche vier Ebenen hoch ist. Abbildung 57 zeigt einen Ausschnitt aus dieser Typhierarchie. Die Feature Codes, die nur aus einem Buchstaben bestehen, werden als direkte Kinder des Wurzeltyps in die Hierarchie eingefügt. Die Codes, die aus zwei Zeichen aufgebaut sind, stellen deren Kinder dar, und so weiter. Insgesamt enthält die aus den Census Class Feature Codes abgeleitete Typhierarchie 258 Typen.

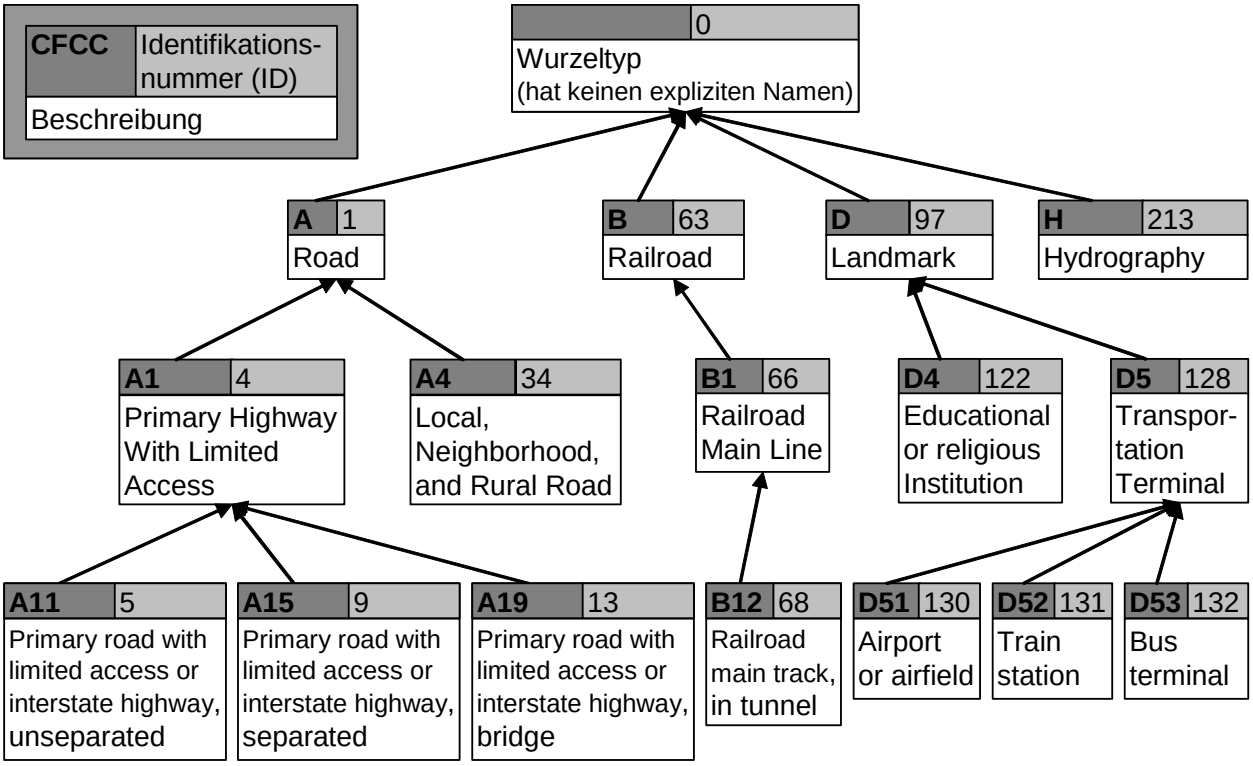


Abbildung 57: Ausschnitt aus der Typhierarchie, die von den Census Feature Class Codes (CFCC) abgeleitet wurde, welche in den TIGER/Line Datensätzen verwendet werden

5.2.5.1 Erwartungswert der Gesamtantwortzeit (EWGAZ)

Die in den Messungen eingesetzten Anfragen bestehen aus je einem Typprädikat und einem räumlichen Prädikat, die beide konjunktiv miteinander verknüpft sind. Aus allen Typen der Typhierarchie wurde eine repräsentative Untermenge ausgewählt, welche die 37 häufigsten Typen enthält. Jeweils einer dieser Typen wird im Typprädikat als der gewünschte Objekttyp eingesetzt. Der Selektivitätsfaktor des Typprädikats bewegt sich damit im Bereich von 0,1% bis 100%. Er bezieht sich auf das Verhältnis der Anzahl der Objekte, die den geforderten Typ oder einen seiner Subtypen aufweisen, zur Anzahl der Objekte im gesamten Datensatz. Für das räumliche Prädikat werden Anfragegebiete mit zehn verschiedenen Selektivitätsfaktoren im Bereich von 0,5% bis 95% generiert. Der Selektivitätsfaktor ist hier definiert als das Verhältnis der Größe des Anfragegebiets zur Größe des Universums des Datensatzes. Jede Kombination von Typ (und damit des Selektivitätsfaktors des Typprädikats) und Selektivitätsfaktor des räumlichen Prädikats führt zu einem Messpunkt in der dreidimensionalen Darstellung in Abbildung 58, wobei die Höhe des Mess-

punkts aus der gemittelten gemessenen Antwortzeit folgt, welche folgendermaßen ermittelt wird.

Für jeden Messpunkt werden 50 verschiedene Anfragegebiete generiert, die alle die gleiche Größe bzw. Fläche haben aber verschiedene Seitenverhältnisse und Positionen. Die Positionen der Anfragegebiete sind gleichverteilt über dem Universum des Datensatzes. Damit werden Verzerrungen durch lokale Streuungen in der Objektverteilung verringert. Für jedes Anfragegebiet werden fünf Messungen vorgenommen und der Mittelwert aus den mittleren drei Antwortzeiten gebildet, um die Verfälschungen durch den Java Garbage Collector zu minimieren. Schließlich wird der Mittelwert aus allen 50 Antwortzeiten berechnet. Dies ergibt einen Messpunkt, der in Abbildung 58 durch einen kleinen Kreis dargestellt ist.

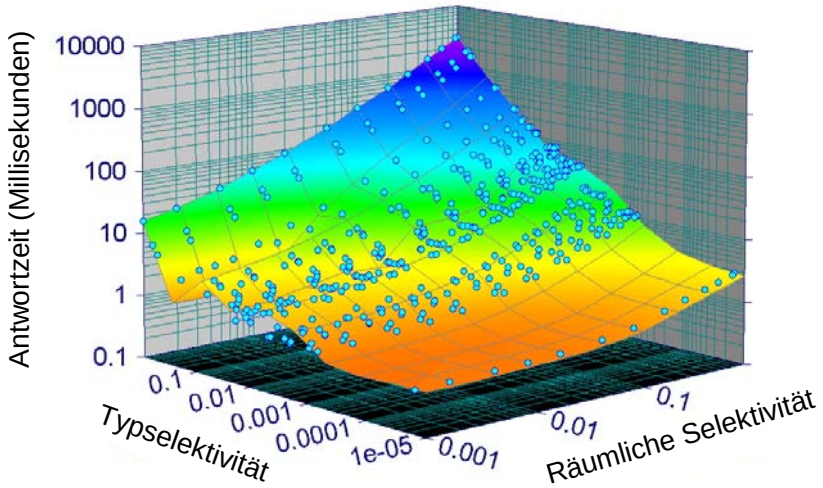


Abbildung 58: Visualisierung der gemessenen Antwortzeiten als Ebene im Raum in Abhängigkeit von der Typselektivität und der räumlichen Selektivität

Da der Austausch einer Indexstruktur gegen eine andere sehr aufwändig ist, da beispielsweise der Index neu aufgebaut werden muss, werden im Folgenden keine einzelnen Messpunkte betrachtet, sondern die Messpunkte werden zu einem Erwartungswert der Anfrageantwortzeit (EWAAZ) zusammengefasst. Für jede Kombination aus Indexstruktur, Einsatzszenario und Datensatz wird ein eigener EWAAZ berechnet. Hierzu werden die einzelnen Messpunkte als Stützpunkte einer stückweise linearen Oberflächenfunktion interpretiert, wobei die Selektivitätsfaktoren des räumlichen Prädikats und des Typprädikats die beiden unabhängigen Variablen darstellen, siehe Abbildung 58. Der EWAAZ wird für jede Kombination aus Indexstruktur, Einsatzszenario und Datensatz berechnet als das gewichtete Integral der Oberflächenfunktion entlang der Achsen der beiden unabhängigen Variablen. Die Integrationsgrenzen ergeben sich aus den minimalen und maximalen Selektivitätsfaktoren, die für jedes Einsatzszenario festgelegt worden sind, siehe Abbildung 49. Die Oberflächenfunktion wird beim Integrieren gewichtet mit dem Kehrwert des Produkts der beiden Selektivitätsfaktoren. Dies wird in Abbildung 59 veranschaulicht. Diese Gewichtung trägt der Beobachtung Rechnung, dass Anfragen, die nur wenige Objekte zurückliefern, wesentlich häufiger gestellt werden als Anfragen, die einen Großteil der Objektmenge auf einmal zurückliefern. Hierdurch soll ein Mittelwert berechnet werden, der das gesamte Selektivitätsspektrum gleichmäßig berücksichtigt.

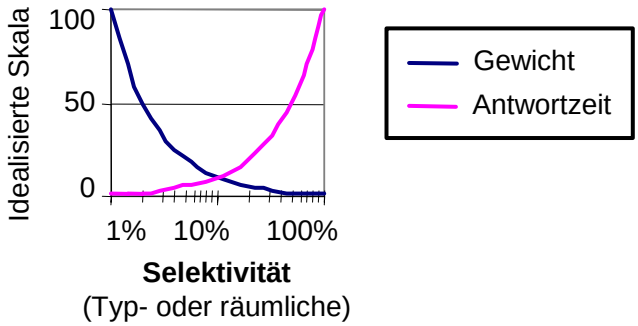


Abbildung 59: Gegenüberstellung von Antwortzeit und deren Gewicht je nach Selektivität

Das Einsatzszenario beeinflusst auch die Häufigkeit mit der Aktualisierungen auftreten. Die jedem Einsatzszenario zugeordnete Aktualisierungsrate kann in Abbildung 49 abgelesen werden. Der Erwartungswert der Gesamtantwortzeit (EWGAZ) setzt sich nun aus der mittleren Zeit, die für das Einfügen eines Objekts benötigt wird und welche mit der Aktualisierungsrate gewichtet wird, und dem Erwartungswert der Anfrageantwortzeit zusammen. Der EWGAZ wird auf diese Weise für jede Kombination von Indexstruktur, Einsatzszenario und Datensatz berechnet. Durch die Aggregation der Anfrage- und der Aktualisierungsleistung in eine einzige Kennzahl können die Gesamtkosten, die beim Einsatz einer bestimmten Indexstruktur entstehen, miteinander verglichen werden.

5.2.5.2 Vergleich der Typabbildungsvarianten des dreidimensionalen Indexes

In diesem Kapitel werden die drei in Kapitel 5.2.4.4 für den „Dreidimensionaler Index“ Ansatz vorgestellten Abbildungsvarianten Gleicher Abstand (GA), Gleiches Gewicht (GG) und Objektverteilung (OV) miteinander verglichen, um die beste Variante für den Vergleich mit den anderen Ansätzen auszuwählen. Die Varianten unterscheiden sich in den Abständen der in den Wertebereich der Typdimension abgebildeten Werte benachbarter Typen. Zusätzlich wird eine vierte Variante untersucht, bei der die Identifikationsnummern der Typen eins zu eins als abgebildeter Wert übernommen werden (1:1), ohne dass die abgebildeten Werte anschließend skaliert werden. Wie man in Abbildung 62 leicht sehen kann ist diese Abbildungsvariante mit Abstand die schlechteste, welche mindestens 37% und bis zu 418% langsamer ist als die schnellste Variante, je nach Einsatzszenario.

Abkürzung	A	B	C	D	E
Wertebereich der Typabbildung (in km)	6,5	165	1.545	14.047	58.831
Vielfaches des vermutet optimalen Wertebereichs	0,006	0,15	1,4	13	54

Abbildung 60: Wertebereiche der Typabbildung

Der Wertebereich der Typabbildung wird in fünf Stufen variiert, welche abkürzend A, B, C, D und E genannt werden. Die jeweilige Größe des Wertebereichs und das Verhältnis zur vermuteten optimalen Größe sind in Abbildung 60 dargestellt. Die vermutete optimale Größe des Wertebereichs wird berechnet als das Produkt aus dem mittleren Abstand der Objekte entlang einer Achse (welcher bei den verwende-

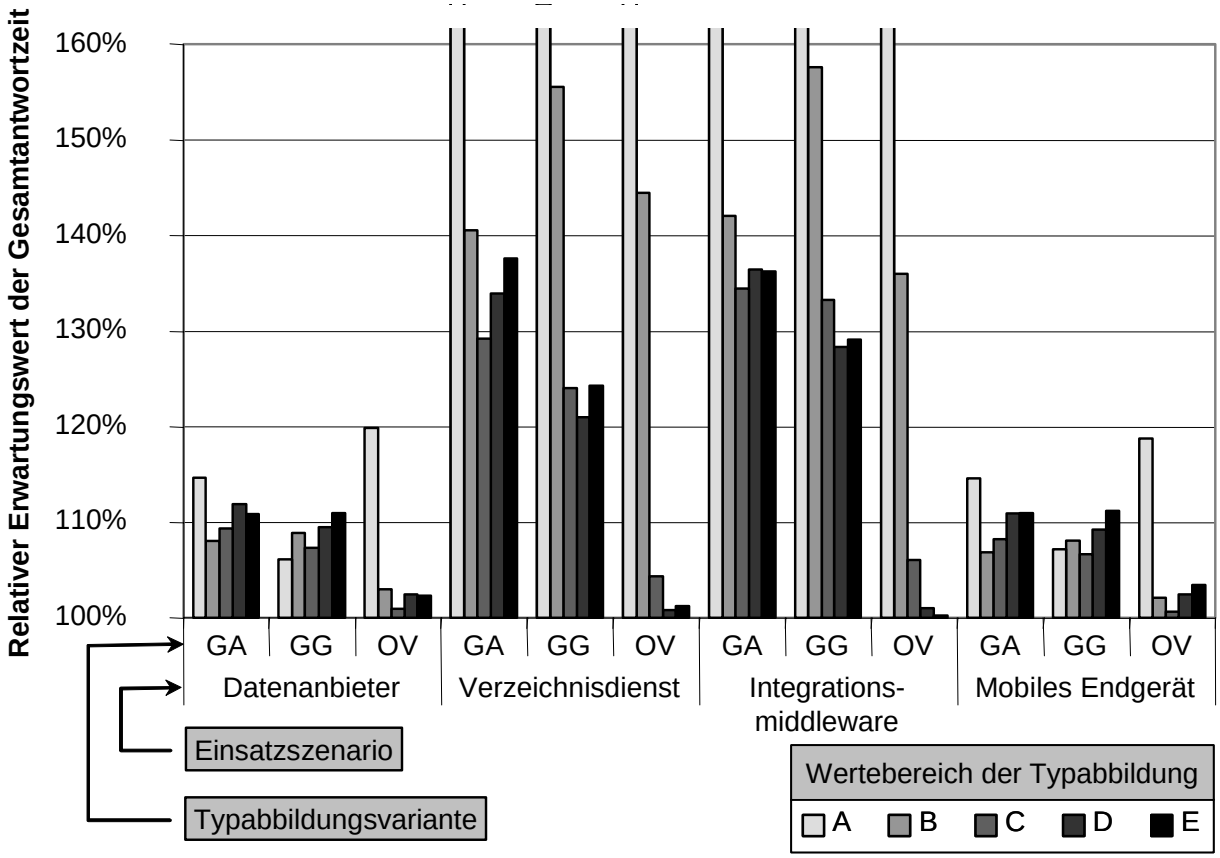


Abbildung 61: Relativer Erwartungswert der Gesamtantwortzeit (EWGAZ) der verschiedenen Typabbildungsvarianten je nach Einsatzszenario und verwendetem Wertebereich der Typabbildung

ten Datensätzen 4200 Meter beträgt) und der Anzahl der Typen in der Hierarchie (258). Damit soll eine gleichmäßige Gruppierung der Objekte in den inneren Knoten des dreidimensionalen Indexes entlang der räumlichen Dimensionen und der Typdimension erreicht werden. Die hier verwendeten Wertebereiche sind aus einem iterativen automatischen Optimierungsprozess hervorgegangen, und repräsentieren jeweils das Optimum einer Iteration, wobei in jeder Iteration der Variationsspielraum begrenzt war. Wie die Ergebnisse in Abbildung 61 zeigen, wurde eine repräsentative Auswahl an Wertebereichen vorgenommen, da überwiegend einer der mittleren Wertebereiche die besten Ergebnisse liefert.

In Abbildung 61 wird der relative Erwartungswert der Gesamtantwortzeit (relativer EWGAZ) dargestellt, welcher wie folgt berechnet wird. Die Messungen werden nach Einsatzszenario und Datensatz gruppiert. In jeder Gruppe wird für jede Kombination aus Typabbildungsvariante und Wertebereich der Typabbildung deren relativer EWGAZ als das Verhältnis des entsprechenden EWGAZ zum minimalen EWGAZ der Gruppe berechnet. Die Säulen in Abbildung 61 stellen jeweils den Mittelwert über alle neun Datensätze dar. Die Berechnung des relativen EWGAZ ermöglicht den Vergleich der Ergebnisse verschieden großer Datensätze sowie die Bildung eines aussagekräftigen Mittelwerts darüber.

Als erstes fällt auf, dass in den Einsatzszenarien Datenanbieter und Mobiles Endgerät der Wertebereich der Typabbildung weit weniger Einfluss auf die Leistung der Anfrageverarbeitungskomponente hat als in den anderen beiden Einsatzszenarien.

Insgesamt ist die wesentliche Erkenntnis aus diesem Vergleich, dass es ausreicht, den Wertebereich der Typabbildung in der Nähe des optimalen Wertebereichs festzulegen. Selbst wenn der Unterschied eine ganze Größenordnung beträgt verschlechtert sich die Leistung dadurch nur um circa 5%. Wenn der Wertebereich jedoch weiter abweicht, dann führt dies zu deutlichen Leistungseinbußen. Leider gibt es keinen eindeutig besten Wertebereich sondern dieser hängt vom Einsatzszenario und der Typabbildungsvariante ab. Gemittelt über alle Szenarien erzielt die Typabbildungsvariante GA die beste Leistung mit dem Wertebereich C, und die Varianten GG und OV arbeiten am effizientesten mit dem Wertebereich D.

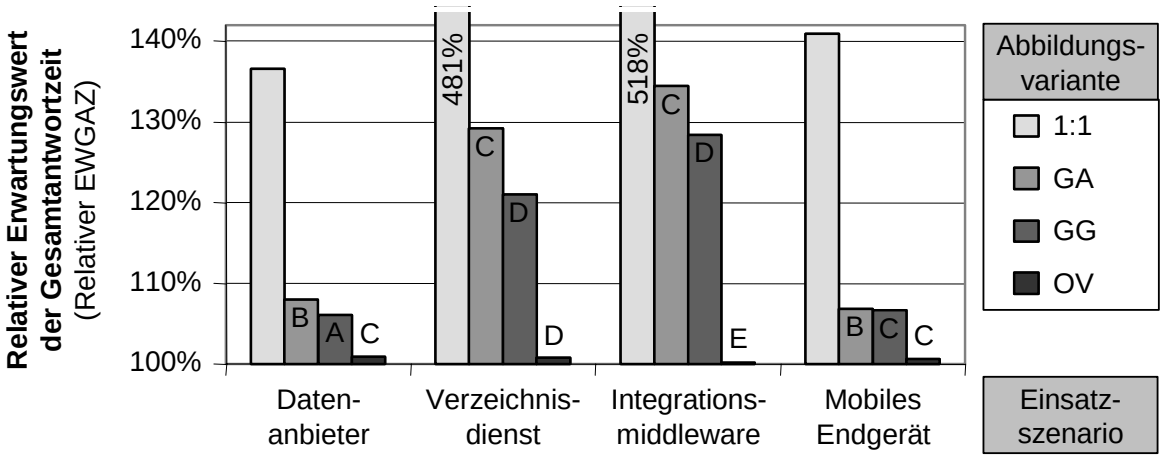


Abbildung 62: Vergleich des relativen EWGAZ für den jeweils besten Wertebereich pro Abbildungsvariante und Einsatzszenario

In Abbildung 62 wird eine Teilmenge der Ergebnisse aus Abbildung 61 dargestellt. Für jede Typabbildungsvariante wird in jedem Einsatzszenario nur jeweils derjenige Wertebereich der Typabbildung gezeigt, der den schnellsten relativen EWGAZ ergibt. In der Abbildung wird der jeweils dargestellte Wertebereich durch dessen Abkürzung am oberen Ende der Säule angezeigt. Zusätzlich wird den drei Typabbildungsvarianten noch die Variante 1:1 gegenübergestellt. Es ist deutlich zu sehen, dass die Variante OV die beste Leistung erbringt. In den Einsatzszenarien Datenanbieter und Mobiles Endgerät ist sie um ungefähr 7% schneller, in den anderen beiden Szenarien sind es mindestens 20%. Man muss jedoch bedenken, dass die Variante OV im Voraus schon zusätzliches statistisches Wissen über die Verteilung der Objekte auf die Typen der Typhierarchie benötigt, um diese Leistung erbringen zu können.

Die beste Alternative, die ohne zusätzliches Wissen auskommt, ist die Variante GG, welche immer zwischen 1% und 8% schneller ist als die Variante GA. Die Variante 1:1 ist immer weit abgeschlagen.

In der weiteren Diskussion wird nur noch die Variante OV berücksichtigt, da diese die beste Leistung erzielt. In jedem Einsatzszenario wird ebenfalls nur derjenige Wertebereich der Typabbildung berücksichtigt, der die beste Leistung erzielt.

5.2.5.3 Indexaufbau

In diesem Kapitel wird der Verwaltungsaufwand der einzelnen Indexstrukturen analysiert. Der Aufwand setzt sich zusammen aus der mittleren Zeit, die für das Einfügen eines Objekts in die Indexstruktur benötigt wird (siehe Abbildung 63) und dem mittleren Speicherverbrauch jedes Objekts (siehe Abbildung 64). In beiden Abbildungen können die insgesamt für jeden Datensatz benötigte Zeit bzw. Speichermenge errechnet werden, indem die dargestellten Werte mit der Anzahl der Objekte im jeweiligen Datensatz multipliziert werden. In den weiteren Betrachtungen wird der Aufwand zum Aktualisieren eines Objekts angenähert durch den Aufwand für das Einfügen eines neuen Objekts.

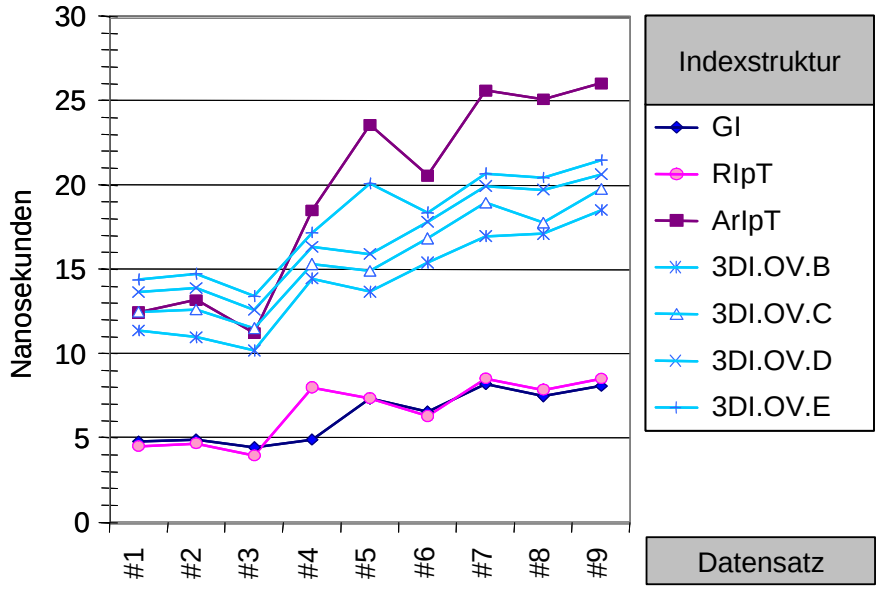


Abbildung 63: Mittlere Dauer des Einfügens eines Objekts in die Indexstruktur

In Abbildung 63 ist zu sehen, dass die Indexstrukturen in den Ansätzen „Getrennte Indexe“ (GI) und „Räumlicher Index pro Typ“ (RIpT) ähnlich schnell aufgebaut werden. Die für jedes Objekt benötigte Zeit wächst nur leicht mit zunehmender Datensatzgröße, so dass beide Ansätze gut skalieren. Der Ansatz „Aggregierter Räumlicher Index pro Typ“ (ArIpT) ist deutlich langsamer als die vorigen beiden Ansätze, und der Abstand wird sogar noch größer bei größeren Datensätzen. Beim kleinsten Datensatz ist der ArIpT Ansatz 2,8 mal langsamer als der GI Ansatz und beim größten Datensatz ist er sogar 3,1 mal langsamer.

Der Ansatz „Dreidimensionaler Index“ (3DI) bewältigt bei kleinen Datensätzen den Indexaufbau ähnlich schnell wie der ArIpT Ansatz. Je nach Wertebereich der Typabbildung ist er zwischen 2,4 und 3,2 mal langsamer als der GI Ansatz. Bei großen Datensätzen ist der 3DI Ansatz klar besser als der ArIpT Ansatz. Hier ist er nur zwischen 2,2 und 2,6 mal langsamer als der GI Ansatz. In Abbildung 63 wird ebenfalls deutlich, dass der Wertebereich der Typabbildung (hier B, C, D oder E) einen erheblichen Einfluss auf die für den Indexaufbau benötigte Zeit hat. Je größer der Wertebereich ist, desto langsamer ist der Indexaufbau. Dies ist darauf zurückzuführen, dass der verwendete MX-CIF Quadtree mehr innere Knoten anlegen muss, um den größeren Wertebereich abzudecken. Die anderen hier nicht dargestellten Typabbildungsvarianten (GA und GG) zeigen ein ähnliches Verhalten.

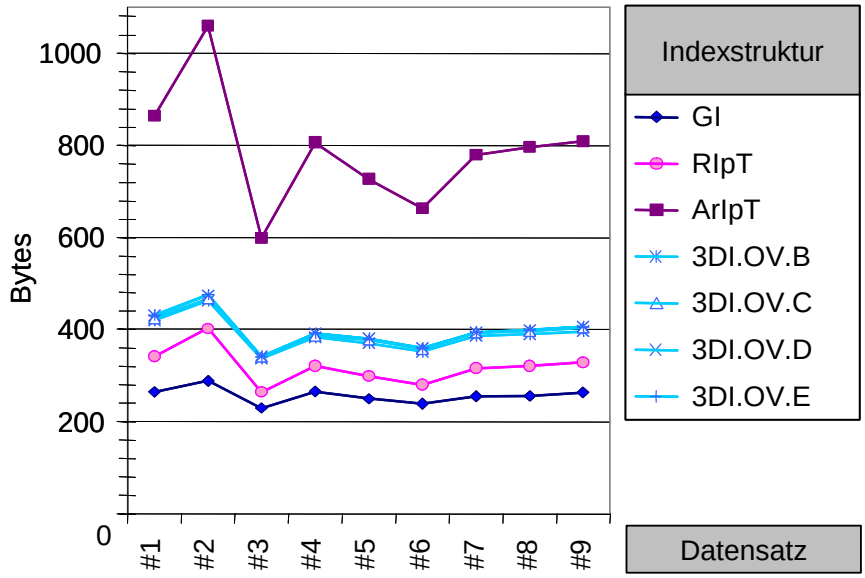


Abbildung 64: Mittlerer Speicherverbrauch pro Objekt

Aus Abbildung 64 kann abgelesen werden, dass die mittlere Größe des Speicherbereichs, der von einem Objekt belegt wird, weitgehend unabhängig von der Größe des Datensatzes ist. Der Speicherbereich umfasst die Geometrie des Objekts, deren kleinstes umschließendes Rechteck, die Identifikationsnummer des Typs des Objekts sowie weitere für die Indexorganisation benötigte Datenstrukturen. Größere Datensätze führen nicht zu einer schlechteren Speicherausnutzung.

Am wenigsten Speicher benötigt der GI Ansatz. Der RIpT Ansatz belegt ungefähr 25% mehr Speicher. Alle Varianten des 3DI Ansatzes benötigen ähnlich viel Speicher. Sie benötigen ungefähr die Hälfte mehr als der GI Ansatz. Weit abgeschlagen ist der ArIpT Ansatz, welcher drei mal so viel Speicher belegt wie der GI Ansatz. Dies entspricht ungefähr dem Grad der Redundanz, den dieser Ansatz herbeiführt. Die hier verwendete Typhierarchie hat vier Ebenen, pro Ebene wird einmal der komplette Datensatz abgelegt. Lokale Unregelmäßigkeiten in der Objektverteilung führen dazu, dass kleinere Indexe vor allem in den unteren Ebenen der Typhierarchie einige innere Knoten einsparen können, so dass insgesamt nur der dreifache Speicherverbrauch verzeichnet wird und nicht der vierfache.

5.2.5.4 Einsatzszenarienspezifische Analyse

In diesem Kapitel wird spezifisch für jedes Einsatzszenario (siehe Kapitel 5.2.3) diejenige Hauptspeicherindexstruktur (siehe Kapitel 5.2.4) ermittelt, mit der die Anfrageverarbeitungskomponente dort die beste Leistung erzielt. Beim Ansatz „Dreidimensionaler Index“ (3DI) wird nur die Typabbildungsvariante berücksichtigt, welche die Gewichtung entsprechend der Objektverteilung (OV) vornimmt. Als Wertebereich der Typabbildung wird jeweils derjenige gewählt, der die beste Leistung erzielt, siehe Abbildung 62.

In Abbildung 65 ist die relative Leistung der vier Ansätze für jeden untersuchten Datensatz getrennt nach den Einsatzszenarien dargestellt. Gleichermaßen wie in Kapitel 5.2.5.2 berechnet sich der relative Erwartungswert der Gesamtantwortzeit

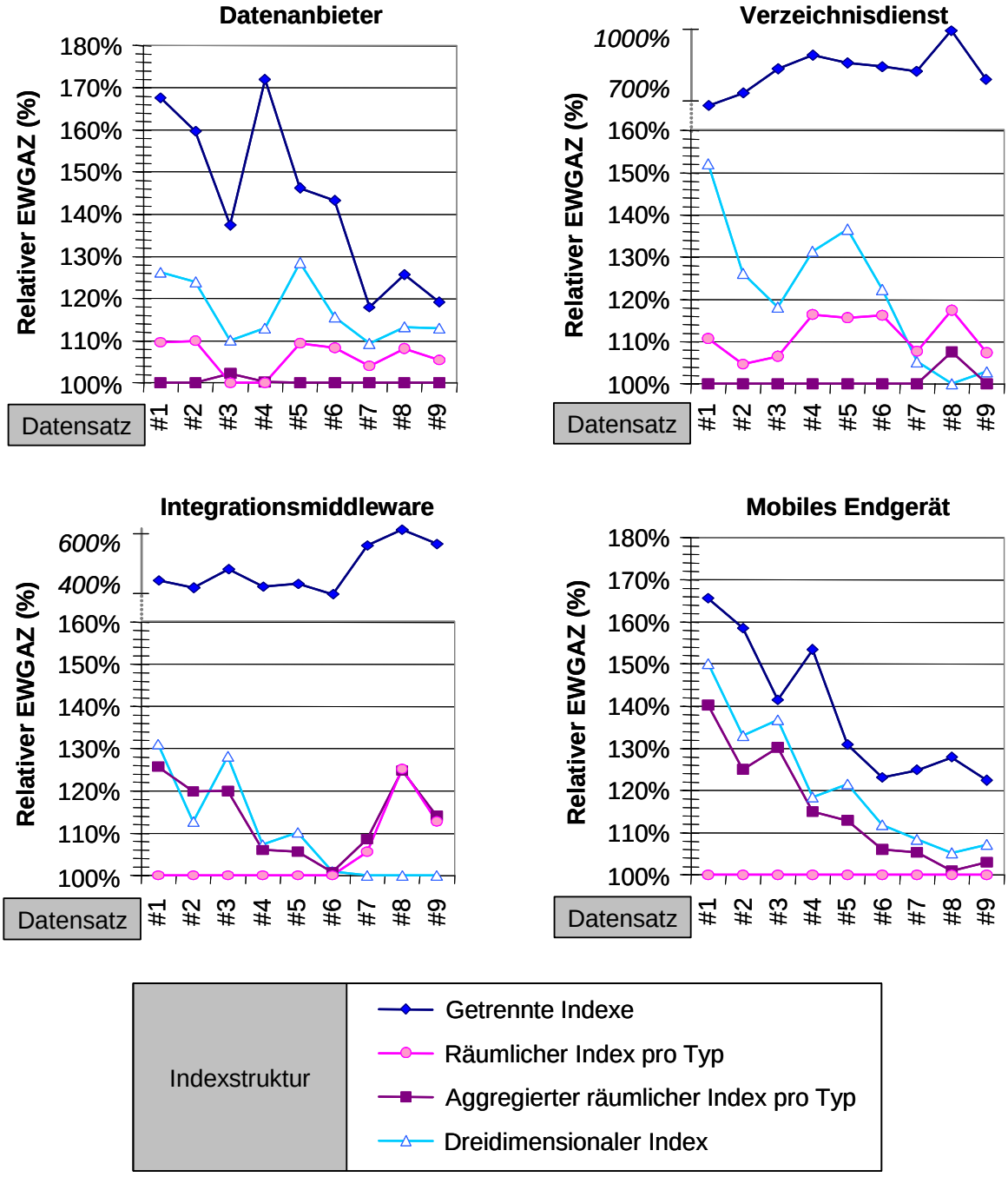


Abbildung 65: Relativer Erwartungswert der Gesamtantwortzeit (Relativer EWGAZ) der verschiedenen Indexstrukturen je nach Einsatzszenario

(relativer EWGAZ) als das Verhältnis des EWGAZ zum minimalen EWGAZ pro Datensatz und Einsatzszenario.

Im Szenario **Datenanbieter** ist der Ansatz „Aggregierter Räumlicher Index pro Typ“ (ArIpT) klar der beste Ansatz. Der Ansatz „Räumlicher Index pro Typ“ (RIpT) ist im Mittel um 6% langsamer. Der Ansatz „Dreidimensionaler Index“ (3DI) belegt den dritten Platz mit einem mittleren Abstand von 17%. Mit zunehmender Datensatzgröße wird der Abstand jedoch geringer. Der Ansatz „Getrennte Indexe“ (GI) zeigt die schlechteste Leistung und liegt mindestens 18% und im Mittel sogar 43%

hinter dem besten Ansatz. Auch hier ist der Abstand geringer bei größeren Datensätzen.

Im Szenario **Verzeichnisdienst** zeigt wiederum der ArIpT Ansatz die beste Leistung. Der RIpT Ansatz ist im Mittel 11% langsamer. Bei kleinen und mittelgroßen Datensätzen ist der 3DI Ansatz wesentlich langsamer (im Mittel um 31%) als der ArIpT Ansatz. Bei großen Datensätzen zeigt der 3DI Ansatz jedoch eine exzellente Leistung, die gleichauf ist mit der des ArIpT Ansatzes. Im Mittel über alle Datensätze reicht es trotzdem nur für den dritten Platz. Der GI Ansatz ist weit abgeschlagen und benötigt im Mittel die achtfache Zeit im Vergleich zum besten Ansatz.

Im Szenario **Integrationsmiddleware** hängt der beste Ansatz von der Größe des Datensatzes ab. Bei kleineren und mittelgroßen Datensätzen ist der RIpT Ansatz am besten und der 3DI Ansatz ist im Mittel 15% schlechter. Bei großen Datensätzen ist es genau anders herum. Eine höhere Aktualisierungsrate würde diesen Wendepunkt zu Gunsten des RIpT Ansatzes verschieben während eine niedrigere Aktualisierungsrate vorteilhaft für den 3DI Ansatz wäre. Der ArIpT Ansatz zeigt bei allen Datensätzen eine Leistung, die vergleichbar mit der des jeweils schlechteren der beiden vorigen Ansätze ist. Der ArIpT Ansatz liegt hier also nur an dritter Stelle. Der GI Ansatz ist wiederum weit abgeschlagen und ist im Mittel 400% langsamer.

Im Szenario **Mobiles Endgerät** schließlich ist der RIpT Ansatz der beste bei allen Datensätzen. Die ArIpT und 3DI Ansätze zeigen eine vergleichbare Leistung bei großen Datensätzen aber sie fallen deutlich zurück bei kleinen und mittelgroßen Datensätzen. Insgesamt ist der ArIpT Ansatz leicht besser als der 3DI Ansatz, er ist im Mittel nur 15% langsamer während der 3DI Ansatz im Mittel 21% langsamer ist als der RIpT Ansatz. In diesem Szenario erreicht der GI Ansatz die Größenordnung der anderen Ansätze, er ist im Mittel nur 39% hinter dem besten Ansatz. Trotzdem ist er bei allen Datensätzen der langsamste Ansatz. Ähnlich wie bei den beiden vorigen Ansätzen ist auch beim GI Ansatz der Abstand zum besten Ansatz bei großen Datensätzen wesentlich kleiner als bei kleinen oder mittelgroßen Datensätzen.

5.2.5.5 Zusammenfassung der Messungen

Überraschenderweise ist der Ansatz „Dreidimensionaler Index“ nur in wenigen Fällen die beste Wahl. Stattdessen liefert dieser Analyse der Ansatz „Räumlicher Index pro Typ“ die besten Ergebnisse. In den Szenarien mit einer hohen Aktualisierungsraten ist er am schnellsten von allen Ansätzen, und in den Szenarien mit einer niedrigen Aktualisierungsrate ist er nur wenig langsamer als der schnellste Ansatz. In letzteren Szenarien zeigt der Ansatz „Aggregierter Räumlicher Index pro Typ“ die beste Leistung, da dort dessen hohe Aktualisierungskosten nur eine untergeordnete Rolle spielen. Dieser Ansatz hat jedoch immer noch den Nachteil eines sehr hohen Speicherverbrauchs. Falls so wenig Speicher wie möglich belegt werden soll, dann ist der Ansatz „Räumlicher Index pro Typ“ zu bevorzugen.

Der Ansatz „Dreidimensionaler Index“ liegt in den meisten fällen nur auf dem dritten Platz. Erst bei großen Datensätzen wird dessen Leistung konkurrenzfähig. Im Szenario Integrationsmiddleware schafft es dieser Ansatz sogar, bei großen Datensätzen schneller als alle anderen Ansätze zu sein. Jedoch wird immer noch ein Administrator benötigt, der einen sinnvollen Wert für den Wertebereich der Typabbildung festlegt, da sonst die Leistung beträchtlich nachlässt. Des Weiteren benötigt

die beste Typabbildungsvariante (Gewichtung nach der Objektverteilung) zusätzliches statistisches Wissen über die Verteilung der Objekte auf die Typen der Typhierarchie.

Bei diesem überraschenden Ergebnis wirken verschiedene Aspekte zusammen. Der dreidimensionale Index enthält immer alle Objekte des Datensatzes, wohingegen in einem typspezifischen Index weniger Objekte enthalten sind, so dass ein solcher Index auch weniger innere Knoten enthält, die traversiert werden müssen. Zusätzlich müssen in einem typspezifischen Index bei der Traversierung nur zweidimensionale Rechtecke verglichen werden, was schneller geht als dreidimensionale Quader zu vergleichen. Die Überschaubarkeit der Typhierarchie führt dazu, dass der Nachteil des Ansatzes „Räumlicher Index pro Typ“, bei dem möglicherweise viele einzelne Indexe durchsucht werden müssen, nicht zum Tragen kommt.

Der Ansatz „Getrennte Indexe“ ist bei Szenarien mit einem niedrigen Selektivitätsfaktor des Typprädikats weit abgeschlagen hinter den anderen Ansätzen. In den anderen Szenarien erreicht er eine Leistung, die in der Größenordnung derjenigen der anderen Ansätze liegt, bleibt jedoch auch hier immer am langsamsten.

Bei großen Datensätzen sind die Leistungsunterschiede weniger stark ausgeprägt als bei kleinen und mittelgroßen Datensätzen. Die Anfrageverarbeitungskomponente kann auf dem Server, auf dem die Messungen durchgeführt wurden, im Mittel zwischen 3500 und 8500 Objekte in 10 Millisekunden verarbeiten, je nach Einsatzszenario und Datensatz. Sie übertrifft damit sogar das in Kapitel 5.2.1 gesteckte Ziel und ist somit darauf vorbereitet, auch zukünftige anspruchsvollere Aufgaben zu bewältigen.

5.2.6 Abgrenzung

Wie schon anfangs angedeutet, können auch andere Informationssysteme wie z. B. OGC Catalogue Services [OGC05a] (OGC CS) oder Grid Metadata Catalog Services [Sing03] (MCS) von der hier diskutierten Anfrageverarbeitungskomponente profitieren. Derzeit bieten Hersteller wie ESRI, Galdos oder Ionic Verzeichnisdienste für Geodaten an, welche die OGC CS Implementation Specification [OGC05a] umsetzen, sowie Server für Geodaten die der Web Feature Service Implementation Specification der OGC [OGC05b] entsprechen. Jedoch konzentrieren sich diese Hersteller darauf, ihre Systeme als Aufsatz auf gängige objektrelationale Datenbanksysteme zu implementieren und somit deren Anfrageverarbeitungskomponente zu verwenden. Deshalb haben die Hersteller wenig Einfluss auf die Arbeitsweise der Anfrageverarbeitung. Insbesondere können sie keine Anfrageverarbeitungskomponente anbieten, die speziell auf die in Kapitel 5.2.2 vorgestellten typischen Anfragen zugeschnitten ist. Forschungsprojekte wie z. B. das GDI NRW Testbed [Bern02] kümmern sich um den Entwurf der übergeordneten Gesamtarchitektur und um den Entwurf der Interaktionsmuster zwischen den einzelnen Komponenten. Die Optimierung der zu Grunde liegenden Datenstrukturen liegt nicht in deren Fokus.

Zhao et al. untersuchen in [Zhao04], wie man am besten einen OGC CS mit Hilfe eines Grid MCS implementiert. Während die Autoren nicht auf die interne Implementierung des Services eingehen, so unterstreicht es doch die Relevanz der hier vorgestellten Anfrageverarbeitungskomponente. Aktuelle Arbeiten im Bereich der Grid MCS befassen sich mit der Verwaltung von erweiterbaren Attributmengen

[Deel04], jedoch nicht mit Indexstrukturen oder mit der Ausnutzung von Klassenhierarchien. Dogac et al. beschreiben in [DKL05], wie Klassenhierarchien in ebXML Verzeichnissen ausgenutzt werden können, ohne dabei jedoch näher auf eine effiziente Implementierung oder eine Unterstützung durch Indexe einzugehen.

Die Hauptspeicherdatenbanksysteme Cloudscape von IBM und TimesTen von Oracle sind rein relationale Systeme und bieten keinerlei Unterstützung für Typhierarchien oder räumliche Indexe. MonetDB bietet geographische Erweiterungen [BQK96], unterstützt jedoch auch keine hierarchischen Beziehungen. Aktuelle Arbeiten in diesem Bereich zielen auf die Minimierung von falschen Sprungvorhersagen des Prozessors [Ross04] und auf die Entwicklung von Datenstrukturen, welche den Aufbau des prozessorinternen Cachespeichers berücksichtigen [MBK00], um den Flaschenhals zum Hauptspeicher zu entschärfen. Diese Arbeiten sind komplementär zu den hier durchgeführten Untersuchungen.

5.2.6.1 Räumliche Indexstrukturen

Ein detaillierter Überblick über die wichtigsten räumlichen Indexstrukturen kann in [GaGü98] gefunden werden. Da der Fokus hier auf der Erstellung einer Anfrageverarbeitungskomponente und nicht auf dem Entwurf einer neuen Indexstruktur liegt, wurde eine wohlbekannte und gängige räumliche Indexstruktur ausgewählt, die mutmaßlich am besten mit der Verarbeitung von realen Datensätzen und hohen Aktualisierungsraten zurecht kommt. Es wurde der MX-CIF Quadtree ausgewählt, da er diese Anforderungen voll erfüllt und leicht zu implementieren ist. Das Ziel ist hier nicht, die beste räumliche Indexstruktur für die vorliegende Problemstellung herauszufinden. Stattdessen wird analysiert in welcher Weise bekannte Indexstrukturen eingesetzt werden können, und wie dabei die räumlichen Dimensionen und die Typdimension kombiniert werden können.

Vaid et al. beschreiben in [VJJS05] mehrere Ansätze, die räumliche Indexe und Textindexe kombinieren, um eine geographische Suchmaschine für das Web zu erstellen. Deren Ziel ist die Erhöhung der Präzision und der Ausbeute bei der Suche. Im Unterschied zur Problemstellung hier enthält der Text einer Webseite wesentlich mehr verschiedene Worte als es Typen in einer Typhierarchie gibt. Des Weiteren sind keine hierarchischen Beziehungen zwischen Worten definiert oder werden in irgendeiner Weise ausgewertet. Schließlich werden dort die Daten in Dateien auf der Festplatte gespeichert. Die dortigen Ergebnisse können also nicht direkt auf das Problem hier übertragen werden. Sie unterstreichen jedoch die Vorteile, die sich aus einer funktions- und anwendungsspezifischen Anfrageverarbeitung ergeben können.

5.2.6.2 Objektorientierte Datenbanken

Im Bereich der Objektorientierten Datenbanken wurden schon einige verwandte Indexierungsprobleme diskutiert [KKD89, LOL92, MuPo97, RaKa95]. Diese Indexe kombinieren den Typ eines Objekts mit einem oder mehreren seiner Attribute. Attribute dürfen jedoch nur einen Wert haben. Im Gegensatz dazu müssen Daten in einer räumlichen Dimension mindestens durch ein Intervall repräsentiert werden können. Aus diesem Grund werden bei objektorientierten Datenbanken nur Indexstrukturen diskutiert, die einzelne Punkte in einem mehrdimensionalen Raum ver-

walten können (engl. point access methods). Demgegenüber weisen die Geometrien der hier verwalteten Objekte eine räumliche Ausdehnung auf, so dass entsprechende räumliche Zugriffsmethoden (engl. spatial access methods) benötigt werden. Zur Bestimmung der geeignetsten Indexstruktur wurden hier umfangreiche Messungen mit realen (räumlichen) Datensätzen durchgeführt. Hierbei wurde auch der Aufwand zur Nachführung von Aktualisierungen mit einbezogen, um dabei die Gesamtkosten, die beim Einsatz verschiedener Indexstrukturen in einem realen Einsatzszenario entstehen, miteinander vergleichen zu können. Des Weiteren können die folgenden nennenswerten Unterschiede identifiziert werden.

Die am meisten verwendete Indexstruktur ist der Class Hierarchy (CH) Index [KKD89], welcher die Objekte primär nach dem indexierten Attribut anordnet und sekundär nach deren Typ. Dieser Index wurde für Szenarien entwickelt, in denen viele Objekte den gleichen Wert im indexierten Attribut haben, was jedoch in einer räumlichen Dimension äußerst selten vorkommt. Konzeptionell ist dieser Index vergleichbar mit dem räumlichen Index des in Kapitel 5.2.4.1 vorgestellten „Getrennte Indexe“ Ansatzes. In [KKD89] werden auch ein Single Class (SC) Index und ein Full Single Class (FSC) Index vorgestellt. Ersterer ist vergleichbar mit dem in Kapitel 5.2.4.2 vorgestellten Ansatz „Räumlicher Index pro Typ“, und letzterer ist vergleichbar mit dem in Kapitel 5.2.4.3 vorgestellten Ansatz „Aggregierter Räumlicher Index pro Typ“. Während in [KKD89] die SC und FSC Indexe als dem CH Index unterlegen eingestuft werden, zeigen die hier durchgeführten Messungen, dass in der hier untersuchten Anwendungsdomäne das Gegenteil der Fall ist.

Der H-Baum [LOL92] fügt Verbindungen zwischen einzelnen SC Indexen ein gemäß der Vererbungsrelation der zugehörigen Typen. Hierdurch entsteht eine einzige relativ komplex zusammenhängende Indexstruktur. Eine theoretische Analyse ergibt jedoch, dass der H-Baum bei der Verarbeitung von Aktualisierungen immer langsamer ist als der Ansatz „Räumlicher Index pro Typ“. Bei der Verarbeitung von Anfragen ist er immer langsamer als der Ansatz „Aggregierter Räumlicher Index pro Typ“. Schließlich ist die Indexstruktur wesentlich komplexer als der in Kapitel 5.2.4.4 vorgestellte Ansatz „Dreidimensionaler Index“. Insgesamt stellt der H-Baum keine nennenswerte Konkurrenz dar und wird deshalb nicht weiter betrachtet.

Der Multikey Type (MT) Index [MuPo97] arbeitet nach einem ähnlichen Prinzip wie der hier vorgestellte Ansatz „Dreidimensionaler Index“ zusammen mit der Typabbildungsvariante Gleiches Gewicht. Jedoch liegt das Hauptaugenmerk in [MuPo97] auf der Berechnung einer optimalen Linearisierung von Typhierarchien, die Mehrfachvererbung enthalten. Die Skalierung der abgebildeten Werte in der Typdimension wird gar nicht angesprochen. Der Vergleich mit der 1:1 Typabbildungsvariante in Kapitel 5.2.5.2 zeigt jedoch, dass dies einen beträchtlichen Einfluss auf Leistung des Indexes hat.

Der Class Division (CD) Ansatz aus [RaKa95] stellt einen Mittelweg zwischen den SC und FSC Indexen dar. Hier lässt sich der Grad der in der Indexstruktur vorhandenen Redundanz kontrollieren. Dadurch lässt sich Anfrageleistung (viel Redundanz) gegen Aktualisierungsleistung (wenig Redundanz) eintauschen und insbesondere auch ein Optimum zwischen beiden Extremen einstellen. Dieser Ansatz kann prinzipiell auch auf die hier vorliegende Problemstellung übertragen werden, jedoch ist der Nutzen des kontrollierten Mittelwegs fraglich. In den Szenarien, in denen die Gesamtleistung von der Leistung bei der Verarbeitung von Aktu-

alisierungen dominiert wird, arbeitet der CD Ansatz langsamer als der dort zum Sieger gekürte Ansatz „Räumlicher Index pro Typ“. In den Szenarien, in denen die Gesamtleistung von der Anfrageleistung dominiert wird, arbeitet der CD Ansatz langsamer als der dort zum Sieger gekürte Ansatz „Aggregierter Räumlicher Index pro Typ“.

5.2.6.3 Objektrelationale Datenbanken

Konzeptionell wird in einem objektrelationalen Datenbanksystem ein Typ als eine Tabelle und ein Attribut als eine Spalte dieser Tabelle abgebildet. In manchen Systemen, wie z. B. PostgreSQL, sind die Daten auch physisch auf diese Weise strukturiert. Ein Objekt wird als ein Tupel in der Tabelle des zugehörigen Typs gespeichert. Anfragen nach Objekten eines bestimmten Typs werden intern auf die Tabellen, die zu Subtypen des geforderten Typs gehören, ausgedehnt. Indexe können auch auf räumlichen Attributen angelegt werden, jedoch nicht auf dem Typattribut, da der Typ eines Objekts nicht explizit gespeichert wird. Auf diese Weise kann in jeder Tabelle ein eigener räumlicher Index angelegt werden, was dem Ansatz „Räumlicher Index pro Typ“ aus Kapitel 5.2.4.2 entspricht.

Das DB2 Datenbanksystem von IBM verfolgt eine andere Strategie. Dort wird eine sogenannte Hierarchietabelle angelegt, die alle Objekte des Datensatzes enthält [Care99]. Diese Tabelle enthält je eine Spalte für jedes Attribut, das irgendwo in der Typhierarchie definiert wird. Eine weitere Spalte speichert explizit den Typ des Objekts. Wenn der Benutzer einen Index auf einer Spalte anlegt, dann wandelt DB2 diesen Index intern in einen zweidimensionalen Index um, der sich aus der angegebenen Spalte und der Typspalte zusammensetzt. Räumliche Daten werden in DB2 mit Hilfe eines Grid Indexes verwaltet. Der kombinierte Indexeintrag setzt sich nun aus der Nummer der Grid Zelle und der Typnummer zusammen, so dass letztendlich dieser Ansatz mit einem CH Index bei objektorientierten Datenbanken vergleichbar ist, siehe Kapitel 5.2.6.2.

Wenn der Typ eines Objekts explizit in einer eigenen Spalte gespeichert wird und wenn eigene Indexe auf diese Spalte und auf die Spalte, welche die räumlichen Daten enthält, angelegt werden, dann können die folgenden Strategien zur Beantwortung von Anfragen zum Einsatz kommen. Der Anfrageprozessor kann beide Indexe nutzen, indem er aus jedem Index diejenigen Tupelidentifikatoren holt, deren Tupel das zugehörige Prädikat erfüllen. Anschließend werden diese beiden Mengen geschnitten, bevor die restlichen Daten der übrig gebliebenen Tupel von der Tabelle eingelesen werden [AsCh75]. Alternativ kann der Anfrageprozessor den selektiveren Index auswählen und damit eine Kandidatenliste erstellen, welche anschließend sequentiell gefiltert wird [SAC+79]. Die hier durchgeführten Messungen haben gezeigt, dass letztere Vorgehensweise, welche auch im Ansatz „Getrennte Indexe“ (siehe Kapitel 5.2.4.1) eingesetzt wird, im Hauptspeicher immer die effizientere ist.

5.2.7 Zusammenfassung

In diesem Kapitel wurde eine hauptspeicherbasierte Anfrageverarbeitungskomponente vorgestellt, welche die Charakteristika typischer Anfragen ausnutzt, die räumliche Prädikate enthalten sowie Prädikate, die den Typ der gesuchten Objekte

festlegen [SGNM06]. Die Anfrageverarbeitungskomponente kann in vielen einzelnen Komponenten eines großen Informationssystems eingesetzt werden. Jede einzelne Anfrageverarbeitungskomponente muss dabei Antwortzeiten in der Größenordnung von zehn Millisekunden erreichen, damit das Gesamtsystem interaktiv genutzt werden kann. Beide Aspekte sprechen für einen Hauptspeicheransatz.

Es wurden vier verschiedene Ansätze untersucht, die von getrennten Indexen auf Typ und Geometrie über verschiedene Kombinationen geometrischer Indexe bis hin zu einer mehrdimensionalen Indexstruktur reichen. Bei der mehrdimensionalen Indexstruktur wurde zudem untersucht, wie die Objekttypen am besten auf Werte in der Typdimension abgebildet werden können. Wie die Experimente gezeigt haben, hängt die beste interne Datenstruktur für die Anfrageverarbeitungskomponente vom Einsatzszenario ab. Deshalb muss die Anfrageverarbeitungskomponente flexibel gestaltet werden, so dass die interne Datenstruktur bei der Installation angepasst werden kann. Dies wird durch den Hauptspeicheransatz erreicht. Gemittelt über alle Einsatzszenarien bietet der Ansatz „Räumlicher Index pro Typ“ die beste Leistung. Ohne eine geeignete Parametrisierung ist der zunächst favorisierte Ansatz „Dreidimensionaler Index“ weit abgeschlagen. Dies ist auch der Ansatz, der bisher in verwandten Arbeiten verfolgt wurde. Durch die hier vorgestellten Typabbildungsvarianten und die Festlegung eines geeigneten Wertebereichs der Typabbildung kann die Leistung erheblich gesteigert werden. Der Wertebereich muss jedoch schon im Voraus festgelegt werden. Die Typabbildungsvariante, welche die Gewichtung der abgebildeten Werte entsprechend der Objektverteilung vornimmt, hat sich als diejenige mit der besten Leistung herausgestellt. Diese benötigt jedoch zusätzliche Statistiken über die Verteilung der Objekte auf die Typen der Typhierarchie. Selbst mit dieser Feinabstimmung ist der Ansatz „Dreidimensionaler Index“ in den meisten Fällen um 10 bis 20 Prozent schlechter als der beste Ansatz, welcher normale Geoindexe in cleverer Weise kombiniert. Hier zeigt sich, dass der simplere Ansatz dem komplexen überlegen ist.

5.3 Automatische Koordinatentransformation

Insbesondere der Ort oder die Grundfläche eines Objekts bilden einen wesentlichen Bestandteil des Umgebungsmodells. Anwendungen müssen die Position von oder die räumlichen Beziehungen zwischen Benutzern, Geräten, Sensoren und Aktoren kennen [DeAb99, Weis91]. Die Ortsinformationen stammen von verschiedenen Sensoren oder externen Informationsquellen und sollen zusammen im lokalen Umgebungsmodell der Anwendung verarbeitet werden. Hierbei ist eine geometrische Repräsentation der Ortsdaten am flexibelsten. Es können räumliche Prädikate wie Enthaltensein oder Überlappung geprüft werden, Vereinigungen oder Schnitte gebildet werden und Distanzen ausgerechnet werden. Aus diesem Grund müssen sowohl die Anwendungen als auch die Integrationsmiddleware in der Anwendungsdomäne der kontextbezogenen Anwendungen mit räumlichen Daten umgehen können.

Den Koordinaten der Geometrien ist immer ein Koordinatensystem zugeordnet, siehe auch Abbildung 66. Es gibt unzählige Koordinatensysteme, und viele davon sind weit verbreitet, wie beispielsweise WGS84 oder NAD83. Anwendungen in Gebäuden nutzen oft ein lokales Koordinatensystem, dessen Ursprung innerhalb

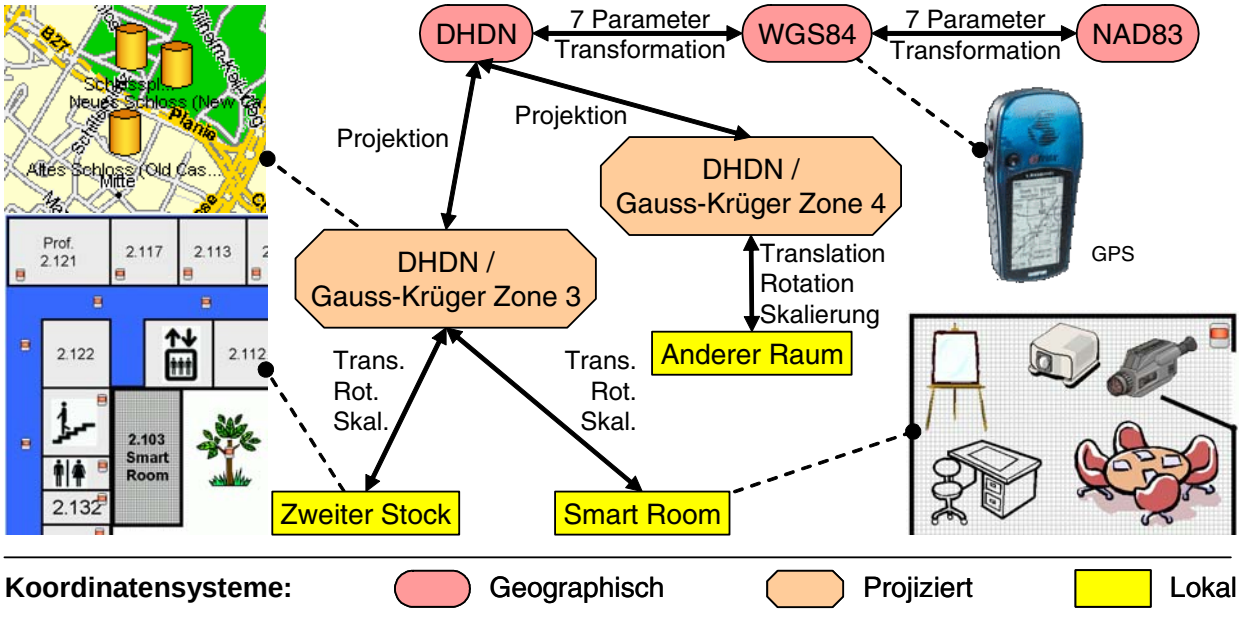


Abbildung 66: Verschiedene Datensätze nutzen unterschiedliche Koordinatensysteme. Diese bilden die Knoten im Transformationsgraph, die Transformationsregeln (z. B. Projektion) bilden die Kanten

des Gebäudes oder eines Raums liegt. Im Freien wird die Position typischerweise mittels GPS (Global Positioning System) bestimmt, welches das WGS84 Koordinatensystem nutzt. Degegen liegen Kartendaten typischerweise in einem projizierten Koordinatensystem vor wie Gauss-Krüger Zone 3. Die EPSG Datenbank [EPSG04] enthält knapp 3000 verschiedene Koordinatensysteme, die weltweit eingesetzt werden. Diese Heterogenität der Koordinatensysteme wird in folgenden Fällen zum Problem:

- Wenn mehrere Anwendungen zusammenarbeiten sollen.
- Wenn verschiedene Positionssensoren wie Infrarotbaken oder GPS integriert werden sollen.
- Wenn Daten aus verschiedenen Quellen, die von verschiedenen Personen oder Organisationen erhoben wurden, zusammen verarbeitet werden sollen.
- Wenn eine Integrationsmiddleware für kontextbezogene Anwendungen erstellt werden soll (dann kommen die drei obigen Punkte zusammen).

Obwohl das Open GIS Consortium mit der Simple Features Specification (SFS) for Corba [OGC99] eine Schnittstelle spezifiziert hat, die grundlegende Operationen auf einfachen Geometrien ermöglicht, die in verschiedenen Koordinatensystemen vorliegen dürfen, gab es zum Zeitpunkt der Entwicklung der Integrationsmiddleware des Nexus-Systems keine adäquate Implementierung davon. Die Java Topology Suite [ViSo04] implementiert die ähnliche SFS for SQL [OGC99a], fordert jedoch, dass alle zu verarbeitenden Geometrien im selben Koordinatensystem vorliegen, welches zudem ein kartesisches sein muss. Im deegree Projekt [deeg04] entstehen einige Referenzimplementierungen aktueller Standards des Open GIS Consortiums, unter anderem der Web Map Service Implementation Specification und der Web Feature Service Implementation Specification. Der freigegebene Quelltext enthält Pakete zur Verarbeitung von Geometrien und zur Koordinatentransfor-

mation. Die einzige implementierte Vergleichsfunktion für Geometrien ist jedoch die Schnittberechnung. Im Spatial Part des SQL99 Standards [ISO99] ist die Funktion ST_Transform definiert, welche Umrechnungen zwischen beliebigen Koordinatensystemen erlauben soll. In der Implementierung durch IBMs Spatial Extender [Davi98] sind jedoch nur Projektionen und keine Geoid-Übergänge möglich. Bei der Verarbeitung wird stets die zweite Geometrie in das Koordinatensystem der ersten Geometrie transformiert, was nicht immer optimal ist.

GeoKlassen

Aus diesem Grund wurden die GeoKlassen entwickelt, welche einen Wrapper um die Java Topologie Suite darstellen, der automatisch die zu verarbeitenden Geometrien in das geeignetste Koordinatensystem transformiert [SHGN04]. Damit können auf einfache und transparente Weise Geometrien in unterschiedlichen Koordinatensystemen verarbeitet werden. Die Umrechnung der Koordinatensysteme erfolgt so selten wie möglich, und die Originalkoordinaten werden so lang wie möglich bewahrt. Die Anwendung bemerkt dabei nichts von den internen Umrechnungsvorgängen.

5.3.1 Geometrietypen und Verarbeitungsfunktionen

Die GeoKlassen implementieren die SFS for Corba [OGC99]. Als Geometrietypen stehen Punkte, Linienzüge und Polygone sowie Gruppen dieser Basistypen zur Verfügung. Punkte können Objekte darstellen, die keine nennenswerte Ausdehnung haben, beispielsweise Benutzer oder kleine Gegenstände wie Sensoren oder Schalter. Linienzüge können Straßen oder berechnete Wegstrecken repräsentieren. Polygone eignen sich für Gebiete wie Gebäudegrundrisse, Räume oder das Empfangsgebiet einer Infrarotbake.

Geometrien können mittels verschiedener Funktionen, die auf Egenhofers Nine Intersection Model [EgHe91] basieren, miteinander verglichen werden auf Gleichheit, Überlappung, Enthaltensein, und so weiter. Neue Geometrien können mit Funktionen wie Vereinigung, Schnittmenge oder Differenz aus vorhandenen Geometrien abgeleitet werden. In den GeoKlassen werden alle Verarbeitungsfunktionen an die entsprechende Implementierung in der Java Topology Suite [ViSo04] weitergeleitet. Zuvor wird, falls notwendig, eine Transformation des Koordinatensystems durchgeführt. Die Java Topology Suite setzt Koordinaten in einem kartesischen Koordinatensystem voraus. Für zukünftige Operationen werden die Originalkoordinaten weiterhin im Geometrieobjekt gespeichert. Die Koordinaten können in drei Dimensionen vorliegen, für die Berechnungen werden jedoch nur die zwei Koordinaten der Ebene berücksichtigt.

Es stehen zwei verschiedene Typen von Koordinatensystemen zur Verfügung. Geographische Koordinatensysteme nähern die Form der Erde durch ein Ellipsoid an. Die Koordinaten geben den Längengrad und den Breitengrad an. Ein projiziertes Koordinatensystem definiert ein ebenes kartesisches Koordinatensystem. Hierfür werden Punkte auf der Oberfläche der Erde durch eine Projektionsvorschrift in die Ebene abgebildet. Beispielsweise benutzt das Gauss-Krüger Zone 3 Koordinatensystem eine transverse Mercator Projektion ausgehend vom geographischen Koordinatensystem DHDN (Deutsches Hauptdreiecksnetz). In Innenräumen wird häufig ein

lokales Koordinatensystem verwendet, welches als Spezialfall eines projizierten Koordinatensystems angesehen werden kann.

5.3.2 Koordinatensystemtransformationen

Im Nexus-System werden derzeit überwiegend das geographische Koordinatensystem WGS84 und das projizierte Koordinatensystem DHDN/Gauss-Krüger Zone 3 benutzt. Die Transformation zwischen beiden Koordinatensystemen läuft in zwei Schritten ab. In beide Transformationsrichtungen muss zunächst in das geographische Koordinatensystem DHDN transformiert werden.

Transformationen wie Projektionen und Geoid-Übergänge können vorwärts und rückwärts durchgeführt werden, dies geschieht jedoch nicht verlustfrei. Abbildung 67 zeigt die Abweichungen der Koordinaten nach einer Vorwärts- und Rückwärts-Transformation vom WGS84 ins DHDN/Gauss-Krüger Zone 3 Koordinatensystem und zurück. Die Kurve zeigt den Abstand zwischen der Ausgangskoordinate und dem Ergebnis nach einer Vorwärts- und Rückwärts-Transformation in Abhängigkeit des Längengrads der Koordinate. Die zweite Kurve zeigt die Ergebnisse wenn als projiziertes Koordinatensystem das DHDN/Gauss-Krüger Zone 4 Koordinatensystem verwendet wird, welches gegenüber dem DHDN/Gauss-Krüger Zone 3 Koordinatensystem einen nach Osten verschobenen Nutzungsbereich aufweist. Abbildung 67 kann wie folgt interpretiert werden. Jeder Transformationsschritt führt unausweichlich zu Ungenauigkeiten, wobei die Abweichungen im besten Fall in der Größenordnung von einem Millimeter liegen. Jedes projizierte Koordinatensystem hat nur einen eingeschränken Nutzungsbereich. Für das DHDN/Gauss-Krüger Zone 3 Koordinatensystem liegt der Nutzungsbereich zwischen 7° 30′ Ost und 10° 30′ Ost. Außerhalb davon nehmen die Abweichungen exponentiell zu. Im Randbereich bewegen sich die Abweichungen noch im Zentimeterbereich und sind damit noch tolerabel. Für das DHDN/Gauss-Krüger Zone 3 Koordinatensystem liegt der Randbereich zwischen 5° 0′ Ost und 7° 30′ Ost sowie zwischen 10° 30′ Ost und 13° 0′ Ost.

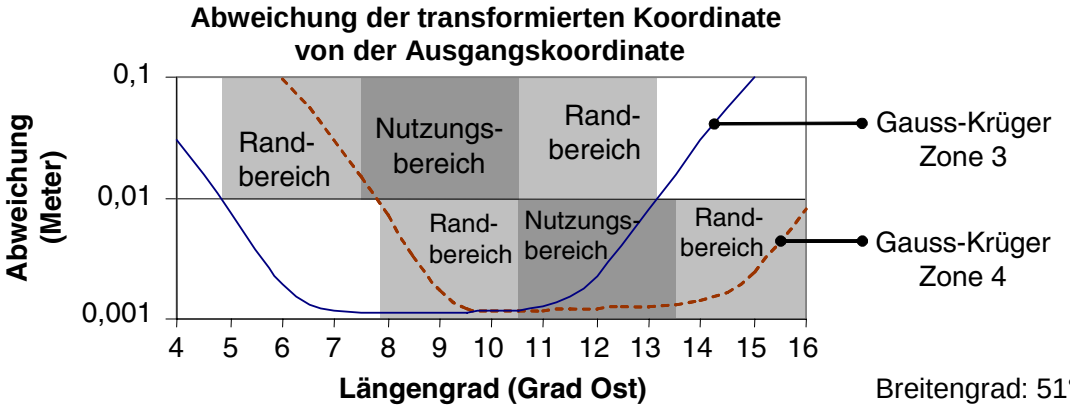


Abbildung 67: Abweichungen der Koordinaten nach einer hin und zurück Umwandlung von WGS84 nach DHDN/Gauss-Krüger Zone 3/4 und zurück nach WGS84 in Abhängigkeit vom Längengrad

Dies zeigt, dass Koordinaten so selten wie möglich transformiert werden sollten, und dass das Zielkoordinatensystem der Transformation sorgfältig ausgewählt

werden sollte. In den GeoKlassen werden Koordinaten erst transformiert, wenn dies unausweichlich ist, z. B. wenn zwei Geometrien verglichen werden sollen. Gleichzeitig werden die Originalkoordinaten weiterhin gespeichert, um bei einer folgenden Transformation von diesen ausgehen zu können.

5.3.3 Bestimmung des gemeinsamen Koordinatensystems

Damit die Java Topology Suite Geometrien verarbeiten oder vergleichen kann, müssen beide im selben kartesischen Koordinatensystem vorliegen. Dies erfordert einerseits die Bestimmung des gemeinsamen Koordinatensystems und andererseits die Transformation der Koordinaten einer oder beider Geometrien in dieses Koordinatensystem.

In der EPSG Datenbank [EPSG04] werden sowohl Koordinatensysteme als auch Transformationsregeln zwischen ausgewählten Paaren von Koordinatensystemen verwaltet. Falls es keine direkte Transformationsregel zwischen zwei Koordinatensystemen gibt, dann können die Koordinatensysteme als Knoten in einem Graph und die existierenden Transformationsregeln als Kanten zwischen diesen Knoten interpretiert werden, siehe Abbildung 66. Die Transformation zwischen den beiden gegebenen Koordinatensystemen setzt sich dann aus der Konkatenation der Transformationen auf dem kürzesten Weg in diesem Graph zwischen den beiden entsprechenden Knoten zusammen.

Lokale Koordinatensysteme können in diesen Graph integriert werden, indem sie in Bezug zu einem bekannten projizierten Koordinatensystem gesetzt werden. Die zugehörige Transformationsregel kann aus einer Rotation, Translation und Skalierung zusammengesetzt sein. Jetzt können Geometrien im lokalen Koordinatensystem des Smart Rooms mit Geometrien im lokalen Koordinatensystem des zweiten Stocks oder mit beliebigen anderen Geometrien, deren Koordinatensystem auf einem Pfad im Graph erreichbar ist, verglichen werden.

Jedem Koordinatensystem wird ein Nutzungsbereich und ein Randbereich zugeordnet, welches jeweils ein Rechteck im globalen WGS84 Koordinatensystem ist. Nach Abbildung 67 liegt der Nutzungsbereich des DHDN/Gauss-Krüger Zone 3 Koordinatensystems zwischen 7°30’ Ost und 10°30’ Ost (und zwischen 47° und 56° Nord wegen der Verschiebung des DHDN Ellipsoids gegenüber dem WGS84 Ellipsoid). Der Randbereich ist um 3°30’ in jeder Richtung größer als der Nutzungsbereich. Im Nutzungsbereich sind die Transformationsfehler minimal, im Randbereich noch tolerabel und außerhalb davon zu groß.

Das gemeinsame Koordinatensystem zweier Geometrien wird wie folgt ermittelt. Wenn beide Geometrien schon im gleichen kartesischen Koordinatensystem vorliegen, dann wird dieses genommen. Andernfalls wird geprüft, ob das Koordinatensystem einer der beiden Geometrien ein kartesisches ist, und ob die andere Geometrie im Randbereich dieses Koordinatensystems liegt. In diesem Fall wird die andere Geometrie in dieses kartesische Koordinatensystem transformiert. Ansonsten müssen für beide Geometrien alle passenden Koordinatensysteme aufgezählt werden solange bis beide Aufzählungen ein gemeinsames Koordinatensystem enthalten. Die Aufzählung erfolgt in den folgenden vier Stufen:

1. Koordinatensysteme, in deren Nutzungsbereich die Geometrie enthalten ist.
2. Koordinatensysteme, deren Nutzungsbereich die Geometrie schneidet.
3. Koordinatensysteme, in deren Randbereich die Geometrie enthalten ist.
4. Koordinatensysteme, deren Randbereich die Geometrie schneidet.

Nach jeder Stufe wird geprüft, ob ein gemeinsames Koordinatensystem entdeckt wurde. Falls mehrere passende Koordinatensysteme gefunden werden, wird dasjenige ausgewählt, das die wenigsten Transformationsschritte erfordert. Falls nach diesen vier Stufen immer noch kein passendes Koordinatensystem gefunden werden konnte, so können die Geometrien nicht zusammen verarbeitet werden. Es wird ein Fehler gemeldet.

5.3.4 Zusammenfassung

Die Möglichkeit, Geometrien in verschiedenen Koordinatensystemen gemeinsam verarbeiten zu können, stellt einen wichtigen Baustein für kontextbezogene Anwendungen dar. Die automatische Transformation der Koordinaten zwischen verschiedenen Koordinatensystemen [SHGN04] vereinfacht sowohl das Nutzen existierender Daten, weil diese nicht vorab in ein passendes Koordinatensystem umgewandelt werden müssen, als auch die Erhebung neuer Daten, beispielsweise durch die Definition eines lokalen Koordinatensystems, das speziell auf das Gebäude zugeschnitten ist, dessen Räume ins Umgebungsmodell integriert werden sollen. Die Geometrien beliebiger Objekte des Umgebungsmodells können nun gemeinsam verarbeitet werden. Dies ist ein wesentlicher Grundpfeiler der Interoperabilität der Umgebungsmodelldaten und der Flexibilität der Integrationsmiddleware.

Für die Darstellung von Räumen in Gebäuden wären unter Umständen auch topologische oder symbolische Lokationsmodelle angebracht, insbesondere weil sie die Erfassung der Daten erheblich vereinfachen würden. Eine entsprechende Erweiterung der GeoKlassen ist in jedem Fall erstrebenswert, bedeutet aber auch, dass die GeoKlassen nicht mehr als reines Geometriepaket betrachtet werden können, sondern dann umfassende Kontextinformationen aus dem Umgebungsmodell für die Verarbeitung topologischer oder symbolischer Geometrien benötigen.

Die Simple Features Specification erlaubt bewusst keine Kreise als Geometrietyp, obwohl diese einige Vorteile bieten: Beispielsweise lassen sich damit elegant Messungenauigkeiten oder Ausbreitungsgebiete repräsentieren. Auch die Enthaltenseinüberprüfung kann sehr effizient berechnet werden. Jedoch wird es sehr kompliziert, wenn aus einem Polygon und einem Kreis eine neue Geometrie berechnet werden soll. Als Lösung könnte ein ähnlicher Ansatz wie bei der Koordinatentransformation verfolgt werden. Der Kreis wird so spät wie möglich in ein regelmäßiges n-Eck, also in ein normales Polygon, umgewandelt.

5.4 Integration domänenspezifischer Funktionalitäten

Eine wesentliche Anforderung an die Integrationsmiddleware der Nexus-Plattform stellt die Integration domänenspezifischer Funktionalitäten in die Anfrageverarbeitung dar. Auf diese Weise sollen oft benötigte Funktionalitäten zur Verfügung gestellt werden, so dass Anwendungen diese nutzen können und nicht selbst implementieren müssen. Auch sollen auf diese Weise ressourcenhungrige Funktionalitäten, die auf einem mobilen Endgerät nicht verarbeitet werden können, nutzbar gemacht werden. Durch die Verarbeitung der domänenspezifischen Funktionalitäten in der Integrationsmiddleware wird auch ein Potential zur Ausnutzung von Synergieeffekten geschaffen. Die Funktionalitäten können zum einen sehr effizient auf die Daten des Umgebungsmodells zugreifen und durch kürzere Antwortzeiten schneller und gezielter auf den Kontext des Benutzers eingehen. Zum anderen können die Ergebnisse der Funktionalitäten zwischengespeichert werden und somit von anderen Benutzern oder anderen Anwendungen genutzt werden ohne neu berechnet werden zu müssen. Domänenspezifische Funktionalitäten gehen weit über die generischen Verarbeitungsmöglichkeiten in traditionellen Datenbanksystemen hinaus. Im weiteren Verlauf dieses Teilkapitels werden einige solcher Funktionalitäten vorgestellt, welche alle schon im Nexus-System genutzt werden können. Die Funktionalitäten sind in die Kategorien Mehrwertdienste (siehe Kapitel 5.4.1) und Ergebnisaufbereitungsoperatoren (siehe Kapitel 5.4.2) aufgeteilt.

5.4.1 Mehrwertdienste

Mehrwertdienste laufen typischerweise in einer Föderationskomponente [NGS+01]. Sie sind jedoch nicht in die Anfrageverarbeitung integriert, sondern sind dieser vorgelagert. Sie bieten Anwendungen eine eigene Schnittstelle, die speziell auf die Funktionalität des Mehrwertdienstes zugeschnitten ist. Mehrwertdienste beziehen die für die Erbringung ihres Dienstes benötigten Daten über die Anfrageverarbeitung der Föderationskomponente. Wenn der Mehrwertdienst und die Anfrageverarbeitung in der selben virtuellen Java Maschine laufen, dann können die Daten direkt in der Hauptspeicherrepräsentation (siehe Kapitel 5.1) übergeben werden und deren Serialisierung kann umgangen werden. Technisch wird hierbei der Servlet Context der Servlet API zur Kopplung von Mehrwertdienst und Anfrageverarbeitung genutzt, da diese jeweils in selbständige Servlets gekapselt sind. Zusätzlich können Mehrwertdienste auch Daten aus anderen Quellen in ihre Verarbeitung integrieren. Im Rahmen des Nexus-Projekts wurden schon einige Mehrwertdienste implementiert, die im Folgenden vorgestellt werden.

Der Kartendienst zeichnet eine Karte und liefert diese als Pixelbild oder als Vektorgrafik zurück. Als Eingabe erwartet er eine AWQL-Anfrage, mit welcher die zu zeichnenden Objekte aus dem Umgebungsmodell beschafft werden sollen, und die Spezifikation der Ausgabe: Grafikformat, Größe des Bildes und die Koordinaten der Eckpunkte.

Der Navigationsdienst berechnet einen Weg. Er benötigt als Eingabe den Start- und den Zielpunkt. Zusätzlich können Zwischenziele angegeben werden. Die Spezifikation sieht ebenfalls vor, dass die zu nutzenden Beförderungsmittel und Navigationspräferenzen vorgegeben werden können, die Implementierung setzt dies jedoch noch nicht um. Als Ausgabe liefert der Navigationsdienst eine Liste von einzelnen

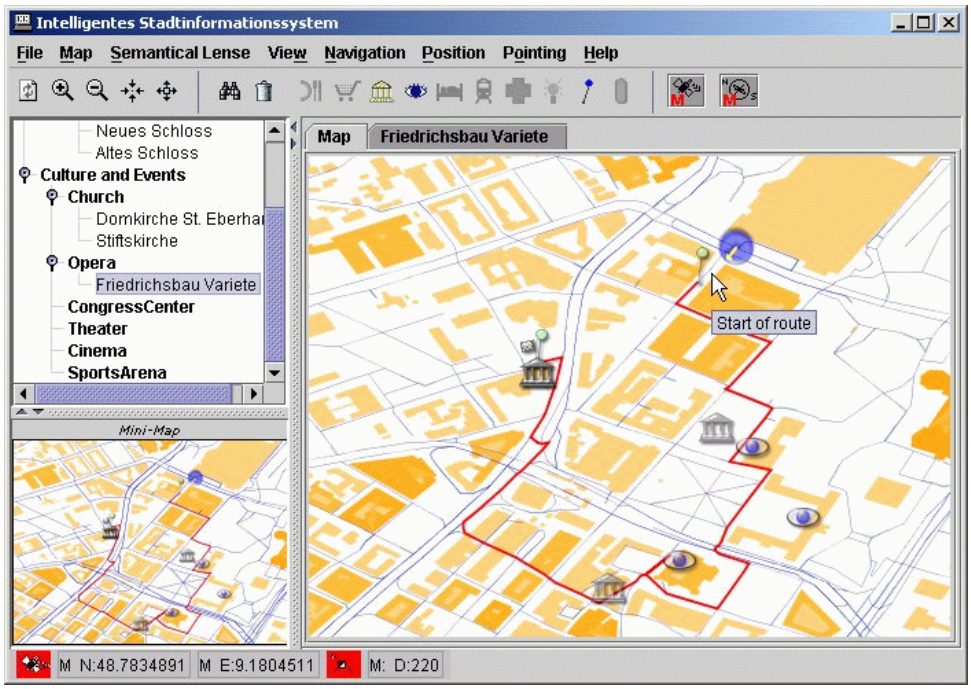


Abbildung 68: Das intelligente Stadtinformationssystem nutzt sowohl den Kartendienst als auch den Navigationsdienst.

Punkten, welche die Zwischenpunkte des zurückzulegenden Weges darstellen. Die Schnittstellen des Kartendienstes und des Navigationsdienstes werden in [BDG+04] genauer beschrieben. Die Ergebnisse beider Dienste sind in Abbildung 68 zu sehen.

Der 3D-Selektionsdienst [Deng06] ermöglicht die Verarbeitung dreidimensionaler Geometrien. Somit können beispielsweise die Räume eines bestimmten Stockwerks in einem mehrstöckigen Gebäude ausgewählt werden. Der Dienst erwartet eine 3D-Anfrage als Eingabe, welche intern zu einer normalen AWQL-Anfrage mit zweidimensionalen räumlichen Vergleichsprädikaten umgewandelt wird. Die von der Föderationskomponente gelieferte Ergebnismenge wird vom Dienst mit den Prädikaten der 3D-Anfrage nochmals gefiltert, um das endgültige Ergebnis zu erzeugen.

Der Fehlersuchedienst ermöglicht einem Entwickler, die Kommunikation zwischen einer Anwendung und der Föderationskomponente zu untersuchen. Hierzu werden Anfragen und Ergebnismengen abgefangen und mit dem in einer weiteren Diplomarbeit entstandenen Nexus Editor [Neum06] grafisch dargestellt. Der Entwickler kann Anfragen abändern, bevor diese von der Föderationskomponente verarbeitet werden, und er kann die zurückgelieferten Ergebnismengen abändern, bevor diese an die Anwendung zurückgeschickt werden. Dies erleichtert die Fehlersuche in der Kommunikation einer Anwendung erheblich. Abbildung 69 zeigt den Fehlersuchedienst in Aktion.

5.4.2 Ergebnisaufbereitungsoperatoren

Ergebnisaufbereitungsoperatoren werden von der Föderationskomponente als Teil der Anfrageverarbeitung ausgeführt [NGS+01]. Die Operatoren können die von den Datenanbietern gelieferten Daten aufbereiten, bevor diese an die Anwendung zurückgeschickt werden. Die Ergebnisaufbereitungsoperatoren erhalten ihre Daten

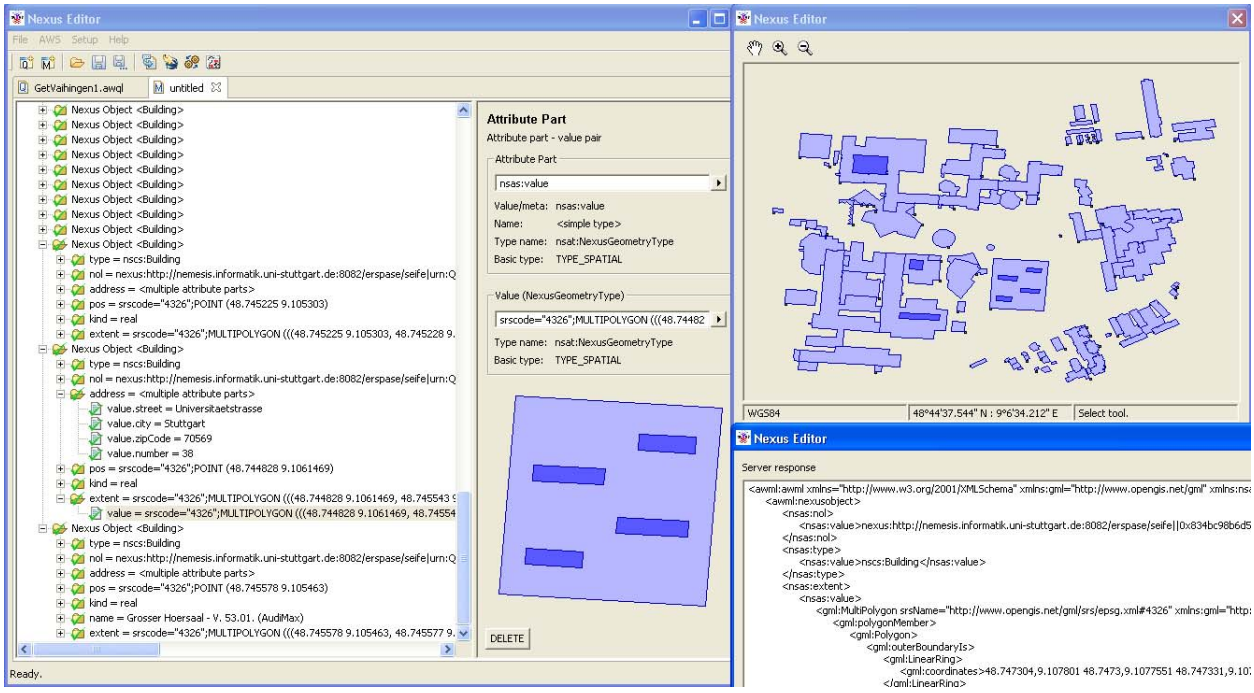


Abbildung 69: Der Fehlersuchedienst fängt Anfragen und Ergebnismengen ab und stellt diese mit dem Nexus Editor grafisch dar.

mittels der in Kapitel 5.1 vorgestellten Hauptspeicherrepräsentation der Umgebungsmodelldaten von der Anfrageverarbeitung. Es ist konzeptionell nicht vorgesehen, dass solche Operatoren auf externe Datenquellen zugreifen. In der Implementierung wird dies jedoch nicht unterbunden.

5.4.2.1 Konzept der Ergebnisaufbereitungsoperatoren

Es gibt zwei Typen von Ergebnisaufbereitungsoperatoren: die verschmelzenden Operatoren und die verarbeitenden Operatoren. Die Schnittstelle beider Typen von Operatoren besteht jeweils aus einer Methode, der Verarbeitungsmethode. Bei verschmelzenden Operatoren hat diese Methode eine Menge von Ergebnismengen als Eingabe und eine einzelne Ergebnismenge als Ausgabe. Jede Eingabemenge repräsentiert die Antwort eines Datenanbieters, so dass die Objekte getrennt nach Datenanbieter verarbeitet werden können. Die Verarbeitungsmethode muss explizit eine neue Ergebnismenge anlegen, und alle Objekte aus den Eingabemengen explizit in die Ausgabemenge übernehmen, wenn diese dort enthalten sein sollen. Verschmelzende Operatoren fassen typischerweise Daten aus mehreren Objekten der Eingabemengen zu einem neuen Objekt zusammen. Durch die explizite Ausgabemenge müssen die Quellobjekte nicht aus den Eingabemengen entfernt werden. Die Eingabemengen sind unveränderlich, so dass hier beispielsweise direkt die in einem Cache gespeicherten Objekte genutzt werden können ohne vorher dupliziert werden zu müssen.

Die Verarbeitungsmethode verarbeitender Operatoren hat ebenfalls eine Menge von Ergebnismengen als Eingabe, jedoch keine Ausgabe. Die Objekte sollen hier an Ort und Stelle verändert werden. Die Eingabemengen sind also gleichzeitig auch die Ausgabemengen. Verarbeitende Operatoren ändern typischerweise nur

bestimmte Aspekte eines Objekts wie dessen Geometrie, so dass das kopieren der Objekte einen unnötigen Aufwand darstellt. Falls Objekte wegfallen oder neue hinzukommen sollen, so müssen die Eingabemengen entsprechend geändert werden. Dies führt dazu, dass Objekte, die aus dem Cache stammen, dupliziert werden müssen, bevor sie an einen verarbeitenden Operator weitergeleitet werden.

An die Verarbeitungsmethoden beider Typen von Operatoren kann zusätzlich ein Parametersatz übergeben werden, mit dem die Verarbeitung konfiguriert werden kann. Die Anwendung kann entweder die Parameter explizit in der Anfrage setzen oder sie kann einen vordefinierten Parametersatz auswählen. Auf diese Weise können zum Beispiel Schwellwerte oder Qualitätseinstellungen angepasst werden.

Die Operatoren werden durch eine Konfigurationsdatei (einem sogenannten deployment descriptor) in die Anfrageverarbeitung integriert. In dieser Datei steht der Name des Operators, dessen Typ und die den Operator implementierende Klasse. Des Weiteren wird dort das Schema des Parametersatzes beschrieben sowie die vordefinierten Parametersätze aufgeführt.

Diese minimale Schnittstellenspezifikation reicht bereits aus, um die in Kapitel 5.4.2.2 vorgestellten Operatoren umsetzen zu können. Das Konzept der Ergebnisaufbereitungsoperatoren kann in folgender Weise erweitert werden.

Die Operatoren sollen die an die Datenanbieter weitergeleiteten Anfragen beeinflussen können. Damit können die Operatoren sicherstellen, dass alle benötigten Attribute angefordert werden, und sie können zusätzliche Kontextdaten aus dem Umgebungsmodell beschaffen. Ein einfacher Ansatz sieht vor, dass alle Typprädikate im *restriction* Teil der Anfrage disjunktiv um eine in der Konfigurationsdatei vorgegebene Liste weiterer Typprädikate ergänzt werden. Somit können zusätzliche Objekte angefordert werden, die denselben Einschränkungen entsprechen wie die ursprünglich von der Anwendung angeforderten Objekte. Zusätzlich kann der *filter* Teil der Anfrage um eine ebenfalls in der Konfigurationsdatei vorgegebene Liste von Attributen ergänzt werden. Falls dieser einfache Ansatz nicht ausreichend ist, so kann ein komplexerer Ansatz verfolgt werden. Dort wird die Schnittstelle der Operatoren um eine Methode ergänzt, die von der Anfrageverarbeitung zwischen Schritt 3 und 4 aufgerufen wird, siehe Kapitel 4.2. Die Methode hat eine Anfrage als Eingabe und eine Anfrage als Ausgabe. Der Operator kann somit die Anfrage der Anwendung komplett umstellen.

Eine andere Erweiterung des Konzepts stellt die Vorgabe von Anforderungen an die Eingabemengen dar. So kann ein Operator fordern, dass eine Eingabemenge in einer bestimmten Sortierung vorliegt, oder dass es einen Index auf einem bestimmten Attributteil gibt. Die Anfrageverarbeitung kann dann entweder die geforderten Eigenschaften selbst herbeiführen, oder sie kann diese Anforderungen an die Datenanbieter weiterleiten. Es ist auch denkbar, dass es mehrere Implementierungen eines Operators gibt, die unterschiedliche Anforderungen an die Eingabemengen haben. Die Anfrageverarbeitung kann dann diejenige Implementierung auswählen, deren Anforderungen sie am leichtesten erfüllen kann, beispielsweise indem der Index des Caches genutzt werden kann oder indem die Sortierung der Objekte schon vom Datenanbieter erledigt wird.

5.4.2.2 Beispiele für Ergebnisaufbereitungsoperatoren

Von den Partnern im Nexus-Projekt wurden schon einige Ergebnisaufbereitungsoperatoren implementiert, welche hier vorgestellt werden sollen. Es gibt einen verschmelzenden Operator, der sich überlappende Straßendatensätze verschmelzen kann [Volz06b]. Dieser wertet hierfür vorab berechnete MRep-Relationenobjekte aus, welche zusammengehörige Straßenobjekte identifizieren, siehe Kapitel 4.4.1.1. Abbildung 70 veranschaulicht das Ergebnis dieses Operators.

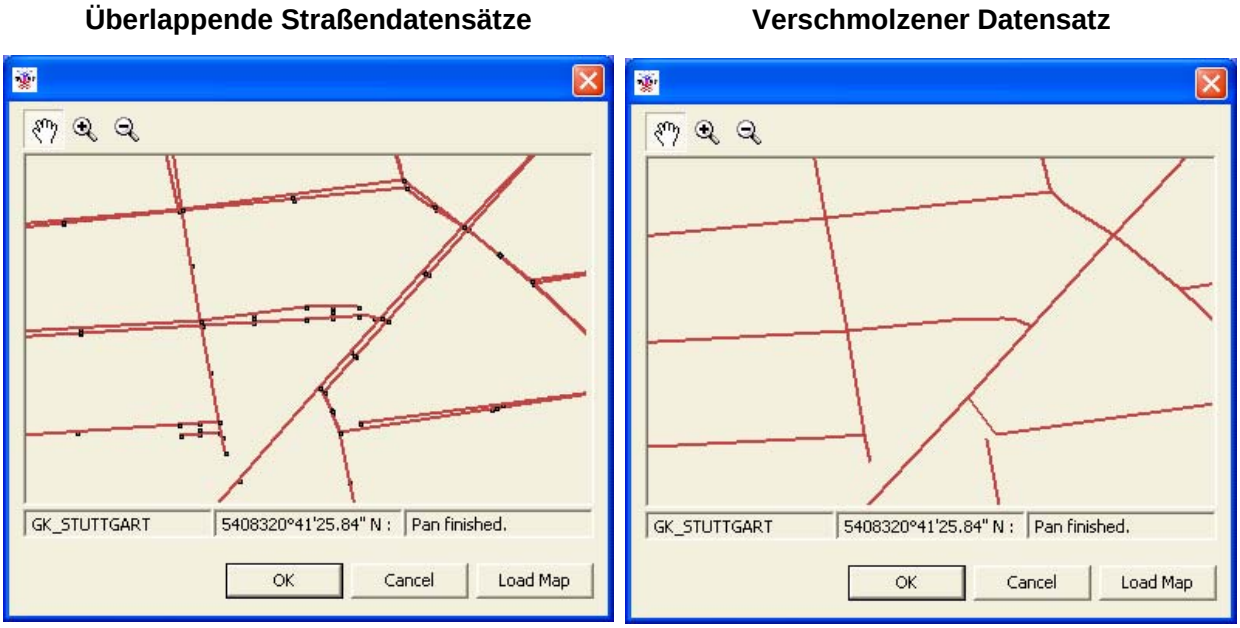


Abbildung 70: Ausgangsdaten und Ergebnis des Ergebnisaufbereitungsoperators, der überlappende Straßendatensätze mit Hilfe von MRep-Relationenobjekten verschmilzt

Bisher gibt es zwei verarbeitende Operatoren, die beide Gebäudegrundrisse vereinfachen. Der Generalisierungsoperator [KaFe06] versucht dabei, den Gesamteindruck des Grundrisses möglichst zu erhalten. Der Simplifizierungsoperator versucht Ähnliches durch weglassen von Zwischenpunkten zu erreichen, ohne die verbleibenden Punkte neu zu berechnen. Die Ergebnisse beider Operatoren sind in Abbildung 71 gegenübergestellt.

5.5 Weitere Maßnahmen

Die in diesem Kapitel beschriebenen weiteren Maßnahmen sind Ideen auf konzeptioneller Ebene zur Weiterentwicklung der Föderationskomponente.

Die Effizienz der Anfrageverarbeitung innerhalb einer Föderationskomponente kann noch weiter gesteigert werden. Insbesondere ist dies der Fall, wenn Ergebnismengen erst gar nicht von entfernten Datenanbietern geholt werden müssen, oder wenn Ergebnisaufbereitungsoperatoren gar nicht ausgeführt werden müssen, weil die jeweiligen Ergebnisse schon vorliegen. Hierfür muss die Föderationskomponente um einen sogenannten Cache erweitert werden. Es eignet sich jedoch kein gewöhnlicher Cache, wie er beispielsweise zum Zwischenspeichern von Webseiten zum Einsatz kommt, da dieser nur Anfragen der Form „Befindet sich Objekt X im

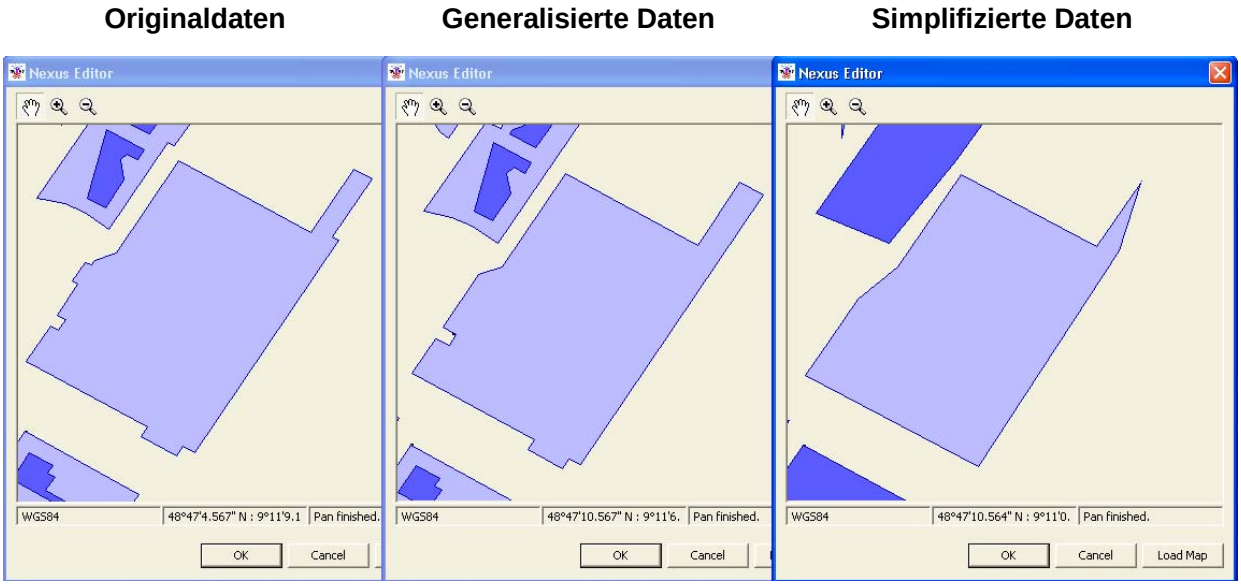


Abbildung 71: Vergleich des Generalisierungsoperators und des Simplifizierungsoperators

Cache?“ beantworten kann. Solche konkreten Anfragen werden nur höchst selten von kontextbezogenen Anwendungen gestellt. Viel häufiger beschreiben die Anfragen deklarativ, an was die Anwendung interessiert ist: „Finde alle Hotels, deren Position im Kartenausschnitt liegt.“. Der Cache muss jetzt Anfragen der Form „Enthält der Cache alle Hotels, deren Position im Kartenausschnitt liegt?“ beantworten. Ein solcher Cache wird deklarativer Cache oder auch semantischer Cache genannt. Zusätzlich soll der Cache auch Anfragen teilweise beantworten können und den dann noch fehlenden Teil identifizieren können. Abbildung 72 verdeutlicht dies an einem Beispiel.

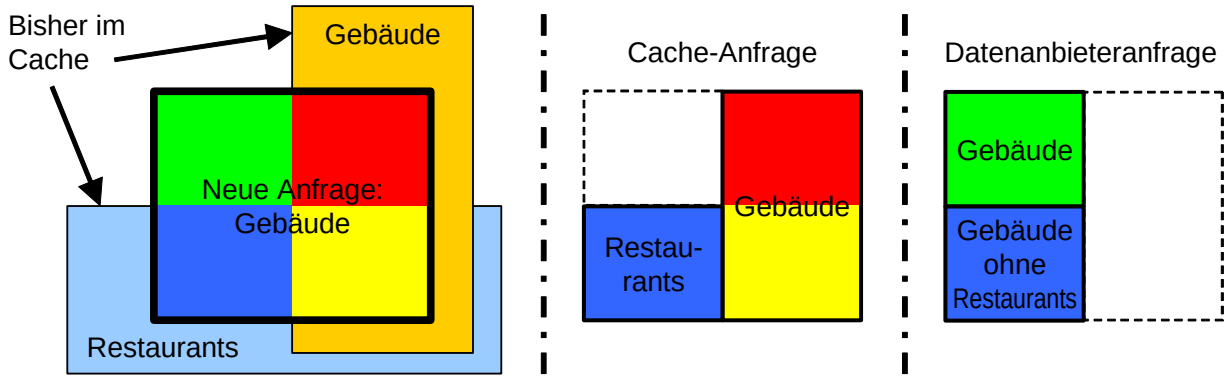


Abbildung 72: Aufteilung einer Anfrage in eine Cache-Anfrage und eine Datenanbieteranfrage

Der Cache enthält die Ergebnisse zweier Anfragen nach Gebäuden und nach Restaurants, die sich teilweise in ihrem Anfragegebiet überlappen. Restaurant sei ein Subtyp von Gebäude, so dass die Restaurants im überlappenden Teil des Anfragegebiets in beiden Ergebnismengen enthalten sind. Eine neue Anfrage fordert alle Gebäude an, die in einem Anfragebiet liegen, das sich mit den Anfragegebieten der beiden vorausgegangenen Anfragen überlappt. Viele gängige Verfahren (siehe Kapitel 5.5.3) versagen hier schon ihren Dienst, da sie nur Anfragen aus dem Cache beantworten können, wenn das Ergebnis der Anfrage vollständig im Cache enthalten ist. Der Cache soll feststellen, dass er die Gebäude im rechten Teil des Anfrage-

gebiets und die Restaurants im unteren Teil des Anfragegebiets bereitstellen kann. Bei den Datenanbietern muss also nur noch nach Gebäuden im oberen linken Teil des Anfragegebiets und nach Gebäuden außer Restaurants im unteren linken Teil des Anfragegebiets gesucht werden. Die Anfrage der Anwendung kann also in eine Cache-Anfrage und in eine Datenanbieteranfrage aufgeteilt werden. Je nach der Lage der Dienstgebiete der Datenanbieter können so ganze Anfragen eingespart werden oder zumindest die Ergebnismengen in ihrem Umfang verringert werden.

Zur Lösung dieser Aufgabe sind verschiedene Ansätze denkbar, welche hier kurz skizziert werden sollen. Hierbei kann wie auch schon in Kapitel 5.2 ausgenutzt werden, dass typische Anfragen räumliche Prädikate und Typprädikate enthalten. Das Hauptproblem stellt die Beschreibung des Cache-Inhalts dar. Im Inhaltsverzeichnis müssen alle Anfragen vermerkt werden, deren Ergebnismengen im Cache zwischengespeichert wurden. Besonders wichtig sind dabei die Prädikate des *restriction* Teils der Anfragen, da sie den Inhalt der Ergebnismenge deklarativ beschreiben. Das Inhaltsverzeichnis muss zudem so organisiert sein, dass effizient bestimmt werden kann, ob eine gegebene Anfrage vollständig oder teilweise aus dem Cache beantwortet werden kann, und welche Datenanbieteranfrage danach noch übrig bleibt. Im Folgenden werden die Ansätze komplexe Geometrien, Hyperquader und Gitterzellen gegenübergestellt (Kapitel 5.5.1). Anschließend werden weitere zu klärende Fragen aufgezeigt (Kapitel 5.5.2). Zum Schluss wird die Eignung der bisherigen Ansätze aus der Literatur erörtert (Kapitel 5.5.3).

5.5.1 Caching-Ansätze

Der Cache wird durch die Anfragen der Anwendungen mit Daten gefüllt. Da die Anwendungen typischerweise auf den mobilen Endgeräten sich bewegender Benutzer ablaufen, sei das folgende Szenario angenommen, welches in Abbildung 73 dargestellt ist. Ein Benutzer läuft durch die Innenstadt und stellt von Zeit zu Zeit Kartenanfragen. Die einzelnen Kartenausschnitte überlappen sich teilweise. Alle Kartenausschnitte sollen im Cache gespeichert werden.

Zur Vereinfachung der Darstellung werden Objekttypen hier ausgeblendet, alle Anfragen zielen auf denselben Typ. Ähnlich wie in Kapitel 5.2 kann ein Typprädikat als ein Bereich in der Typdimension repräsentiert werden. Ebenfalls ausgeblendet werden Projektionen. Es wird angenommen, dass immer vollständige Objekte angefordert werden. Projektionen können ebenfalls in einer eigenen Dimension repräsentiert werden, wobei jedem Attribut ein Bereich in dieser Dimension zugeordnet wird. Je nach der Auswahl der Attribute kann es sein, dass eine Projektion durch mehrere nicht zusammenhängende Bereiche repräsentiert werden muss.

Komplexe Geometrie

Bei diesem Ansatz wird der Inhalt des Caches durch eine komplexe Geometrie beschrieben. Wenn das Ergebnis einer neuen Anfrage dem Cache hinzugefügt wird, so wird als neue beschreibende Cache-Geometrie die Vereinigung aus der bisherigen Cache-Geometrie und der Geometrie des Anfragegebiets berechnet. In [Enge04] wurde gezeigt, dass dieser Ansatz prinzipiell funktioniert. Die Datenanbieteranfrage kann durch eine Subtraktion der Cache-Geometrie vom Anfragegebiet rein geometrisch berechnet werden. Es hat sich aber ebenfalls herausgestellt, dass die

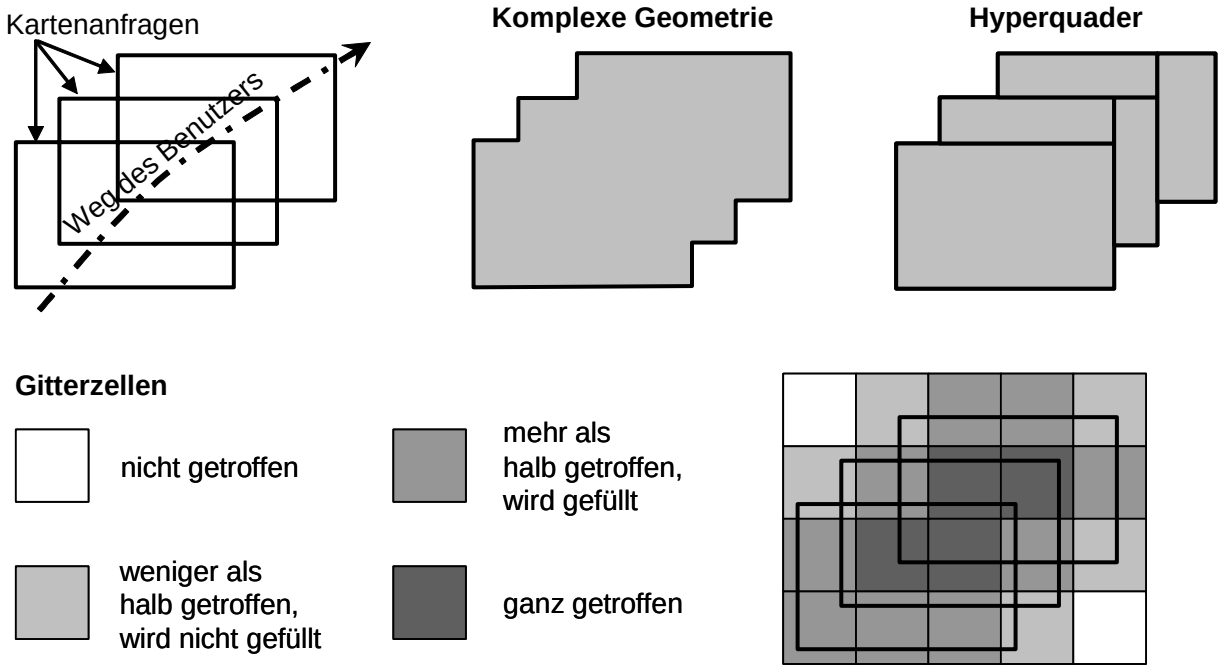


Abbildung 73: Beispielszenario und Aufbau des Inhaltsverzeichnisses der drei Caching-Ansätze

beschreibende Cache-Geometrie schon nach kurzer Zeit aus sehr vielen Eckpunkten besteht, so dass sämtliche Berechnungen und Vergleiche sehr aufwändig werden. Dieser Ansatz scheidet damit aus.

Hyperquader

Der Hyperquaderansatz beschreibt den Inhalt des Caches durch eine Menge an Hyperquadern. Im Beispiel in Abbildung 73 wird dies durch Rechtecke dargestellt. Jedes Attribut, das in einem Prädikat einer Anfrage auftaucht wird durch eine eigene Dimension im Hyperquader repräsentiert. Wenn in einer Anfrage ein Hyperquaderattribut nicht auftaucht, so füllt der Hyperquader in dieser Dimension den kompletten Bereich aus. Ein Hyperquader entspricht genau einem Term einer in die disjunktive Normalform transformierten Anfrage. Wenn das Ergebnis einer neuen Anfrage dem Cache hinzugefügt wird, dann wird die Anfrage aufgeteilt in den Teil, der schon im Cache vorhanden ist, und den Teil, der dort noch fehlt. Der fehlende Teil wird dem Cache-Inhaltsverzeichnis hinzugefügt, nachdem der fehlende Teil in Hyperquader aufgeteilt wurde. Im Beispiel in Abbildung 73 wird der fehlende Teil einer neuen Anfrage in zwei Rechtecke aufgeteilt. Allgemein können auch mehr Rechtecke entstehen, beispielsweise wenn der Benutzer seine Bewegungsrichtung ändert und nach links oben läuft. Im allgemeinen Fall mit beliebig vielen Dimensionen wird die Aufteilung in Hyperquader entsprechend komplexer. Es ist noch kein effizienter Algorithmus bekannt, mit dem zu einer gegebenen Anfrage bestimmt werden kann, ob diese ganz oder teilweise vom Cache beantwortet werden kann. Die Ermittlung der Datenanbieteranfrage ist ähnlich schwierig. Immerhin können schnell die zu betrachtenden Hyperquader gefunden werden, wenn diese in einem räumlichen Index organisiert sind. Damit die Anzahl der Hyperquader im Cache-Inhaltsverzeichnis überschaubar bleibt, können benachbarte Hyperquader zusammengefasst werden. Ein entsprechender Algorithmus muss jedoch auch erst noch entwickelt werden.

Gitterzellen

Der Gitterzellenansatz umgeht die Komplexität des Hyperquaderansatzes, indem der Datenraum in viele disjunkte Zellen aufgeteilt wird, welche in einem Gitter angeordnet sind. Das Gitter hat genauso viele Dimensionen wie die Hyperquader im vorigen Ansatz. Die Gitterzellen können entweder ganz leer oder ganz gefüllt sein. Wenn eine Gitterzelle ganz gefüllt ist, bedeutet das, dass alle Objekte, die zu diesem Bereich des Umgebungsmodells gehören, im Cache gespeichert sind. Die Aufteilung einer Anfrage ist sehr einfach. Es werden alle Gitterzellen aufgezählt, deren Bereich den durch die Prädikate der Anfrage definierten Bereich scheidet. Wenn eine Gitterzelle gefüllt ist, wird ihr Bereich zur Cache-Anfrage hinzugefügt. Wenn sie leer ist, kommt er zur Datenanbieteranfrage. Bei leeren Zellen kann zusätzlich noch entschieden werden, ob diese Zelle komplett gefüllt wird, so dass die Objekte bei zukünftigen Anfragen verwendet werden können, oder ob nur der Teil der Zelle von den Datenanbietern angefordert wird, der tatsächlich für die aktuelle Anfrage benötigt wird. Im letzteren Fall kann dieser Teil des Ergebnisses nicht in den Cache aufgenommen werden. Als eine vielversprechende Strategie erscheint, Zellen ganz zu füllen, deren Bereich mehr als die Hälfte mit der Anfrage überlappt. Der Nachteil dieses Ansatzes besteht darin, dass möglicherweise sehr viele einzelne Zellen betrachtet werden müssen. Dadurch, dass die meisten Anfragen nicht exakt auf die Gitterstruktur passen entsteht entweder zunächst unnötiger Zusatzaufwand, wenn halb getroffene Gitterzellen ganz von den Datenanbietern angefordert werden, oder es können die von einem Datenanbieter gelieferten Objekte nicht weiterverwendet werden, da der Ansatz mit halb gefüllten Gitterzellen nichts anfangen kann. Für die Bestimmung der Zellengröße und die Aufteilung des Gitters liegen noch keine Erfahrungswerte vor. Durch die Ausdehnung der Objekte kann es vorkommen, dass ein Objekt mehr als einer Gitterzelle zugeordnet wird. In gleicher Weise kann dies bei Hyperquadern auch eintreten.

5.5.2 Offene Fragestellungen

Bis die oben beschriebenen Ansätze tatsächlich in der Föderationskomponente eingesetzt werden können, müssen noch die folgenden Fragen geklärt werden. Die Fragen sind gleichermaßen für alle Ansätze relevant.

- Welche Cache-Inhalte sollen aus dem Cache entfernt werden, wenn Platz benötigt wird? Beim komplexe Geometrie Ansatz müsste die Geometrie beschnitten werden. Bei den anderen Ansätzen können die Objekte einzelner Hyperquader oder Gitterzellen aus dem Cache entfernt werden.
- Sollen die Originaldaten der Anbieter oder die an die Anwendung ausgelieferte integrierte Ergebnismenge gespeichert werden? Das Speichern der Originaldaten hat den Vorteil, dass diese als Eingabe für verschmelzende Ergebnisaufbereitungsoperatoren verwendet werden können. Nachteilig dabei ist ein höherer Verwaltungsaufwand. Das Speichern der integrierten Ergebnismenge hat den Vorteil, dass die Daten im Cache direkt an Anwendungen ausgeliefert werden können. Unklar ist dabei, wie neue oder nicht mehr verfügbare Datenanbieter gehandhabt werden können, und wie Änderungen in den Originaldaten eines Anbieters im Cache nachvollzogen werden können.

- Wie können die im Cache gespeicherten Daten aktuell gehalten werden? Wie bemerkt eine Föderationskomponente, dass sich die Originaldaten geändert haben? Wie kann ein Datenanbieter potentiell viele Föderationskomponenten effizient über Änderungen auf dem Laufenden halten? In [Vrho04] werden verschiedene pessimistische und optimistische Verfahren vorgestellt und ihre Eignung für die Konsistenzhaltung eines Föderationscaches diskutiert.
- Wie werden die gespeicherten Objekte verwaltet? Die Objekte können entweder direkt bei der jeweiligen Inhaltsbeschreibung gespeichert werden, oder sie können separat verwaltet werden. Der erste Ansatz hat den Vorteil, dass ausgehend von den Datenstrukturen, die zur Aufteilung der Anfrage in eine Cache-Anfrage und eine Datenanbieteranfrage verwendet werden, direkt auf die zugehörigen Objekte zugegriffen werden kann, ohne nach ihnen suchen zu müssen. Problematisch sind hierbei Objekte, die wegen ihrer Ausdehnung mehreren Gitterzellen oder Hyperquadern angehören. Beim zweiten Ansatz gibt es kein Duplikatproblem, dafür muss für die Objekte eine separate Zugriffsstruktur angelegt und traversiert werden.

Das grundlegende Cache-Konzept lässt sich auch für andere Aufgabenstellungen einsetzen, wobei weitere Fragen gelöst werden müssen.

- Wie lässt sich das Cache-Konzept auf das Zwischenspeichern der Ergebnisse beliebiger Operatoren, also auch der Ergebnisaufbereitungsoperatoren, erweitern? Wie lassen sich dort Änderungen der Ausgangsdaten bemerken und in den zwischengespeicherten Daten nachführen?
- Wie lässt sich das Cache-Konzept auf mobilen Endgeräten einsetzen? Wie können dabei Rundsendeverfahren eingesetzt werden, um effizient die Caches vieler mobiler Endgeräte bei möglichst geringem Bandbreitenverbrauch konsistent zu halten?
- Wie können mehrere Föderationskomponenten mit benachbarten Zuständigkeitsgebieten, die beispielsweise mit den Basisstationen benachbarter Mobilfunkzellen verbunden sind, kooperieren? Wie können diese sich über die gespeicherten Cache-Inhalte absprechen, so dass sie gemeinsam einen größeren Ausschnitt aus dem Umgebungsmodell abdecken können.
- Lässt sich das Cache-Konzept auch auf den Verzeichnisdienst übertragen? Können dabei Anfragen in ähnlicher Weise den Datenbestand eines Datenanbieters beschreiben? Kann dadurch die Effizienz des Verzeichnisdienstes oder der föderierten Anfrageverarbeitung gesteigert werden?

5.5.3 Abgrenzung

In der Literatur existieren schon eine ganze Reihe von Ansätzen für semantische oder deklarative Caches. Prädikatbasierte Verfahren [DFJ+96, KeBa96] beschreiben den Cache-Inhalt mittels den Prädikaten der Anfragen mit denen der Cache befüllt wurde. Die Nutzbarkeit für neue Anfragen wird mittels Subsumptionstests [GSW96, LaYa85, RoHu80] geprüft. Es werden jedoch keine Verfahren vorgestellt, um diese typischerweise mit exponentiellem Aufwand verbundene Berechnung zu beschleunigen. Ähnlichkeiten bestehen auch mit dem Problemfeld des Beantwortens von Anfragen mittels materialisierter Sichten [GoLa01, Hale01]. Die Problem-

stellung dort ist jedoch viel allgemeiner und komplexer, so dass die bisherigen Lösungen viel generischer sind und aufwändige Berechnungen durchführen.

Anfrageschablonenbasierte Verfahren [APTP03] ermöglichen einen effizienten Vergleich einer gegebenen Anfrage mit den im Cache gespeicherten Anfragen. Bislang müssen dabei jedoch die gespeicherten und die zu überprüfenden Anfragen die gleiche Struktur aufweisen. Eine teilweise Nutzung vorhandener Cacheinhalte ist nicht vorgesehen. Der oben vorgestellte Hyperquaderansatz soll die Effizienz der Anfrageschablonen mit der Flexibilität der prädikatbasierten Verfahren kombinieren und zusätzlich die teilweise Beantwortung von Anfragen aus dem Cache erlauben.

In [DRS+98] wird ein komplett gegensätzlicher Ansatz verfolgt indem eine mehrdimensionale Domäne in einzelne Zellen aufgegliedert wird, die jeweils komplett oder gar nicht im Cache vorgehalten werden. Die Arbeit fokussiert jedoch insbesondere auf die Verarbeitung von OLAP Aggregaten entlang mehrerer Dimensionen. Der dortige Ansatz ist sehr ähnlich zum hiesigen Gitterzellenansatz, jedoch besitzen hier die Objekte eine Ausdehnung und können nicht durch Punkte im Raum repräsentiert werden.

Aktuell werden Datenbank-Caching Verfahren entwickelt, die insbesondere auf die Unterstützung von Verbundanfragen ausgerichtet sind. In [ABK+03, HäBü04a, HäBü04b] wird die Nutzbarkeit des Caches für eine gegebene Anfrage mittels einer Prüfanfrage ermittelt. Das Verfahren ist speziell auf Gleichheitsverbunde ausgerichtet. Eine offensichtliche Übertragbarkeit auf Bereichsanfragen ist derzeit nicht ersichtlich. In [HSS04] wird ein Trie als Datenstruktur zur Verwaltung der auf einem mobilen Endgerät zwischengespeicherten Anfrageergebnisse eingesetzt. Das Verfahren ist jedoch speziell auf Verbundanfragen ausgerichtet und behandelt den effizienten Zugriff auf Bereichsprädikate nachrangig.

Im Bereich mobiler Anwendungen wurden einige Ansätze vorgestellt, welche die prädikatbasierten Verfahren um die Betrachtung von Projektionen und um spezielle auf mobile Nutzer zugeschnittene Ersetzungsstrategien erweitern [LLS99, ReDu00, RDK03]. Der Vergleich einer gegebenen Anfrage mit den gespeicherten Anfragen wird mengentheoretisch durchgeführt, ohne hierfür spezielle effiziente Datenstrukturen zu beschreiben. In [LLS02] wird ein zellenbasiertes Rundsendeverfahren zur Verbreitung von Informationen an mobile Endgeräte vorgestellt. Möglicherweise kann die dortige Methode zur Bestimmung der optimalen Zellengröße auf den Gitterzellenansatz übertragen werden. Proaktives Caching [HXW+05] transportiert einzelne Indexknoten auf das mobile Endgerät und soll damit sowohl Gebiets- als auch Nachbarschaftsanfragen verarbeiten können. Die Nützlichkeit im Zusammenhang mit Aktualisierungen bleibt offen, ebenso die Einsetzbarkeit in föderativen Umgebungen, wenn ein globaler Index auf Grund der zu großen Datenmenge nicht mit vertretbarem Aufwand erstellt werden kann.

Bislang wurde kein semantischer Cachingansatz vorgestellt, der sowohl eine hohe Trefferquote und Nutzungsrate erreicht als auch in effizienter Weise viele gespeicherte Anfragen verwalten kann, wie dies in der Föderationskomponente benötigt wird. Gerade in der Anwendungsdomäne der kontextbezogenen Anwendungen können die Charakteristika der typischen Anfragen ausgenutzt werden, und räumliche Indexe oder sogar kombinierte räumliche und Typindexe, wie in Kapitel 5.2 beschrieben, zur Organisation der Cache-Inhalte und zur Suche nach den für eine

gegebene Anfrage passenden Cache-Fragmenten genutzt werden. Hierdurch können die Leistungsprobleme der mengentheoretisch oder prädikatenlogisch arbeitenden Verfahren umgangen werden.

5.6 Bewertung

In diesem Kapitel wurde die Anfrageverarbeitung innerhalb der Föderationskomponente erörtert und verschiedene Konzepte zur Erfüllung der an die Föderationskomponente gestellten Anforderungen (siehe Kapitel 2.2) vorgestellt. Die automatische Koordinatentransformation und die verschmelzenden Ergebnisaufbereitungsoperatoren ermöglichen die *Integration* der Daten mehrerer Anbieter. Die einheitliche Datenstruktur der Hauptspeicherrepräsentation sorgt für Heterogenitäts*transparenz* und die verschmelzenden Ergebnisaufbereitungsoperatoren sorgen durch das Zusammenfassen der über viele Anbieter verstreuten Daten zu einem Objekt für Verteilungs*transparenz*. Die Ergebnisaufbereitungsoperatoren allgemein sowie die Mehrwertdienste bieten die Möglichkeit zur *Datenaufbereitung*. Die Mehrwertdienste erfüllen zudem die Anforderung nach *Mehrwertdiensten* in der Integrationsmiddleware. Die *Effizienz* der Anfrageverarbeitung kann durch spezielle auf die orts- und typbezogene Anfrageverarbeitung zugeschnittene Datenstrukturen sowie durch Caching gesteigert werden.

Die Hauptspeicherrepräsentation ist auf die effiziente Speicherung der Daten und den effizienten Zugriff darauf ausgelegt. Die Klassen dienen als Schnittstelle für Ergebnisaufbereitungsoperatoren und Mehrwertdienste. Sie können selbst Anfragen verarbeiten, um Objekte auszuwählen oder Teilmengen zu bilden, und nutzen dabei die orts- und typbezogene Anfrageverarbeitung. Diese nutzt die Charakteristika typischer Anfragen aus, die meist räumliche Prädikate und Typprädikate enthalten. Der Ansatz mit einem eigenen räumlichen Index pro Typ liefert die besten Ergebnisse mit realen Geodaten. Er hat gegenüber dem komplexeren 3D-Index den Vorteil, dass er nicht parametrisiert werden muss. Es hat sich gezeigt, dass der 3D-Index ohne eine adäquate Skalierung in der Typdimension keine konkurrenzfähigen Leistungen liefert.

Die Automatische Koordinatentransformation ermöglicht die transparente gemeinsame Verarbeitung von Geometrien, die in unterschiedlichen Koordinatensystemen vorliegen. Die notwendigen Konvertierungen erfolgen automatisch im Hintergrund. Dies erleichtert die Interoperabilität verschiedener Datenanbieter und die Zusammenarbeit verschiedener Anwendungen erheblich.

Die Integration domänenspezifischer Funktionalitäten in die Anfrageverarbeitung entlastet Anwendungen auf mobilen Endgeräten, da häufig benötigte Funktionalitäten nicht selbst entwickelt und auch nicht selbst ausgeführt werden müssen. Die Ausführung in der Föderationskomponente ermöglicht einen effizienten Zugriff auf das Umgebungsmodell. Zusätzlich können Synergien mit anderen Benutzern oder Anwendungen ausgenutzt werden, wenn beispielsweise Ergebnisse wiederverwendet werden. Mehrwertdienste sind der Anfrageverarbeitung vorgelagert und können ihre Ergebnisse in eigenen Formaten zurückliefern. Ergebnisaufbereitungsoperatoren sind in die Anfrageverarbeitung integriert und ändern im Vorübergehen bestehende Objekte des Umgebungsmodells oder erzeugen neue.

Beim Caching werden von Datenanbietern gelieferte Ergebnismengen zwischengespeichert, um damit zukünftige Anfragen schneller beantworten zu können. Der Cache soll auch zur teilweisen Beantwortung von Anfragen genutzt werden können, um die Datenanbieter zu entlasten.

Die Hauptspeicherrepräsentation nutzt die orts- und typbezogene Anfrageverarbeitung, welche speziell die Charakteristika typischer Anfragen von kontextbezogenen Anwendungen zur Steigerung der Effizienz ausnutzt. Die automatische Koordinatentransformation wird speziell zur Verarbeitung heterogener Geodaten benötigt. Mehrwertdienste und Ergebnisaufbereitungsoperatoren bieten die Möglichkeit, Domänenwissen und spezifische, in der Anwendungsdomäne benötigte, Funktionalitäten in die Anfrageverarbeitung zu integrieren. Das Caching kann durch den Einsatz von räumlichen Indexen und durch das Ausnutzen der Eigenschaften typischer Anfragen wesentlich vereinfacht werden und damit effizienter gestaltet werden. Insgesamt wurde die Problemstellung der Anfrageverarbeitung innerhalb der Föderationskomponente umfassend erörtert und alle Anforderungen konnten erfüllt werden. Zusätzlich wurden effiziente Verfahren zur Beschleunigung der internen Verarbeitung vorgestellt.

6

Zusammenfassung und Ausblick

In diesem Kapitel werden die wesentlichen Beiträge der vorliegenden Arbeit zusammengefasst und es wird auf mögliche Ergänzungen der vorgestellten Konzepte und Erweiterungen der Föderationskomponente hingewiesen.

6.1 Zusammenfassung

In dieser Arbeit wurde die Verarbeitung ortsbezogener Anfragen in lose gekoppelten föderierten Systemen eingehend untersucht und im Prototyp des Nexus-Systems umgesetzt. Nachdem die durch das Nexus-Projekt vorgegebenen Randbedingungen erhoben wurden (Kapitel 2), wurde die Architektur des Nexus-Systems speziell nach den sich daraus ergebenden Anforderungen konzipiert (Kapitel 3). Die Anfrageverarbeitung in der Föderationskomponente wurde einerseits aus dem Blickwinkel der Interaktionen zwischen den verschiedenen Komponenten des Nexus-Systems (Kapitel 4) und andererseits aus dem Blickwinkel der internen Verarbeitung in der Föderationskomponente betrachtet (Kapitel 5). Hierdurch wurden die Grundprobleme gelöst, wie sich verteilte Umgebungsmodelle von lose gekoppelten Datenanbietern föderieren lassen, wie ortsbezogene Anfragen in einem solchen Umfeld verarbeitet werden können, und wie sich der Kontextbezug und weitere Spezifika der Anwendungsdomäne zur Steigerung der Leistung und Effizienz ausnutzen lassen.

In Kapitel 2 wurde zunächst die Idee des Nexus-Systems beschrieben. Anschließend wurden die Anforderungen erhoben, die an eine Föderationskomponente gestellt werden, welche das Konzept des globalen Umgebungsmodells umsetzen soll. Schließlich wurde das dem Umgebungsmodell zu Grunde liegende Datenmodell eingeführt, welches insbesondere Objekte mit Mehrfachattributen und Mehrfachtypen ermöglicht.

In Kapitel 3 wurde die Basisarchitektur des Nexus-Systems beschrieben. Es wurden die beteiligten Komponenten und deren Aufgaben vorgestellt, sowie die Zusammenarbeit mit der Integrationsmiddleware bzw. der darin enthaltenen Föderationskomponente erläutert. Anschließend wurde das föderative Integrationsprinzip auf Basis der Lokal-als-Sicht Föderationsmethode erörtert, wodurch die Autonomie der Datenanbieter und die Anpassung der Anfrageverarbeitung auf dynamisch wech-

selnde Konfigurationen von Datenanbietern sichergestellt werden können. Es folgen eine Beschreibung des Datenaustauschformat, der Anfragesprache und des Verzeichnisdienstes sowie ein Vergleich des Nexus-Systems mit alternativen Architekturansätzen.

In Kapitel 4 wurde zuerst die Semantik von Anfragen auf Objekte mit optionalen Attributen und Mehrfachattributen formal definiert, sowie die Dualität der verschiedenen Anfragesemantiken bei negierten Vergleichsprädikaten hergeleitet. Anschließend wurde die Vorgehensweise bei der Verarbeitung von Gebietsanfragen in der Föderationskomponente erläutert. Für die Verarbeitung von föderierten Nachbarschaftsanfragen wurden mehrere Ansätze vorgestellt und experimentell verglichen. Die Ansätze unterscheiden sich in den vorausgesetzten Fähigkeiten der Datenanbieter und in den zur Auswahl der relevanten Datenanbieter verwendeten Statistiken. Durch die Verwendung einer geschätzten globalen Objektedichte und durch moderate Parallelisierung konnten erhebliche Leistungszuwächse erzielt werden. Für eine konsistente Behandlung von Mehrfachrepräsentationen mussten die bisheringen Anfrageverarbeitungskonzepte erweitert werden. Nach einer systematischen Analyse aller möglichen Verteilungen der für die Objektauswahl relevanten Informationen auf mehrere Datenanbieter konnte daraus ein Algorithmus abgeleitet werden, der sicherstellt, dass eine Ergebnismenge weder Duplikate noch irrtümliche Treffer enthält, sofern die Mehrfachrepräsentationen über MRep-Relationenobjekte zugeordnet werden können. Wenn die geforderten Konsistenzgarantien abgeschwächt werden, kann auch der durch diesen Algorithmus bedingte Zusatzaufwand eingeschränkt werden. Das Kapitel wird abgeschlossen mit einer Diskussion der Möglichkeiten, die sich durch die Zusammenarbeit benachbarter Föderationskomponenten ergeben, beispielsweise indem diese ihre gespeicherten Cache-Inhalte koordinieren.

In Kapitel 5 wurden zuerst die intern bei der Verarbeitung der Anfragen verwendeten Datenstrukturen zur Repräsentation von Anfragen und Ergebnismengen vorgestellt. Anschließend wurde eine speziell auf die Charakteristika typischer Anfragen zugeschnittene Indexorganisation erläutert und experimentell mit alternativen Ansätzen verglichen. Der initial favorisierte Ansatz mit einer dreidimensionalen Indexstruktur hat den Nachteil, dass er erst mit einer adäquaten Parametrisierung konkurrenzfähige Resultate erzielt. In den Experimenten konnte sich der Ansatz mit getrennten räumlichen Indexen für jeden Objekttyp durchsetzen. Danach wurde eine Klassenbibliothek vorgestellt, welche die gemeinsame Verarbeitung von Geometrien gestattet, die in unterschiedlichen Koordinatensystemen vorliegen. Die notwendigen Transformationen werden transparent im Hintergrund durchgeführt, wodurch die Zusammenarbeit und Interoperabilität verschiedener Datenanbieter und Anwendungen erheblich vereinfacht wird. Mittels Mehrwertdiensten und Ergebnisaufbereitungsoperatoren können domänenspezifische Funktionalitäten in die Föderationskomponente verlagert werden. Dadurch wird die Anwendungsentwicklung vereinfacht und die Ausführung der Funktionalitäten kann von einem direkten Zugriff auf das Umgebungsmodell profitieren. Die Effizienz der Föderationskomponente kann weiter gesteigert werden, wenn sie von Datenanbietern gelieferte Ergebnismengen zwischenspeichert, um bei der Verarbeitung zukünftiger Anfragen eventuell auf die Interaktion mit Datenanbietern ganz verzichten zu können. Die Grundlagen für einen entsprechenden semantischen Caching-Ansatz wurden als Abschluss dieses Kapitels präsentiert.

Tabelle 7 zeigt, wie die in Kapitel 2.2 aufgestellten Anforderungen durch die in dieser Arbeit entwickelten Konzepte erfüllt werden konnten.

Tabelle 7. Anforderungen und ihre Umsetzung

Anforderung	umgesetzt durch
Autonomie	• Berücksichtigung eines Datenanbieters nach Anmeldung beim Verzeichnisdienst (Kapitel 3.4) • Ermittlung der relevanten Datenanbieter mittels des Verzeichnisdienstes in den Verfahren zur Verarbeitung von Gebietsanfragen (Kapitel 4.2) und föderierten Nachbarschaftsanfragen (Kapitel 4.3)
Integration	• Anfrageverarbeitungsverfahren (Kapitel 4.2 bis 4.4) sammeln Daten bei relevanten Datenanbietern ein • Unterschiedliche Koordinatensysteme der Geometrien werden automatisch angepasst (Kapitel 5.3) • Verschmelzende Ergebnisaufbereitungsoperatoren können Mehrfachrepräsentationen zusammenfassen (Kapitel 5.4.2)
Transparenz	• Schematransparenz durch das globale Schema des Umgebungsmodells (Kapitel 2.3) • Heterogenitätstransparenz durch die Basisarchitektur (Kapitel 3.1), das einheitliche Datenaustauschformat (Kapitel 3.2) und die einheitliche Anfragesprache (Kapitel 3.3) sowie durch die Hauptspeicherrepräsentation als weiteres Schnittstellenformat (Kapitel 5.1) • Orts- und Namenstransparenz durch die deklarative Anfragesprache (Kapitel 3.3) im Zusammenspiel mit dem Verzeichnisdienst (Kapitel 3.4) und durch die Zurückverfolgbarkeit eines Objekts zu seinem Datenanbieter mittels des NOL (Kapitel 2.3) • Verteilungstransparenz durch die Anfrageverarbeitungsverfahren (Kapitel 4.2 bis 4.4) und durch die verschmelzenden Ergebnisaufbereitungsoperatoren (Kapitel 5.4.2)
Anfragetypen	• Gebietsanfragen (Kapitel 4.2) • Nachbarschaftsanfragen (Kapitel 4.3)
Informationsmaximierung	• Anfrageverarbeitungsverfahren (Kapitel 4.2 und 4.3) bestimmen aktuell relevante Datenanbieter mit Hilfe des Verzeichnisdienstes (Kapitel 3.4) • Daten zugehöriger Mehrfachrepräsentationen werden mit Hilfe von MRep-Relationenobjekten identifiziert und hinzugeladen (Kapitel 4.4)

Tabelle 7. Anforderungen und ihre Umsetzung

Anforderung	umgesetzt durch
Datenaufbereitung	• Mehrwertdienste erzeugen Ergebnisse in eigenen Datenformaten (Kapitel 5.4.1) • Ergebnisaufbereitungsoperatoren verändern Objekte oder erzeugen neue Objekte (Kapitel 5.4.2)
Mehrwertdienste	• Mehrwertdienste (Kapitel 5.4.1)
Effizienz	• Die Effizienz wurde experimentell bei der Verarbeitung föderierter Nachbarschaftsanfragen (Kapitel 4.3) und bei der orts- und typbezogenen Anfrageverarbeitung im Hauptspeicher (Kapitel 5.2) bewertet • Die Zusammenarbeit benachbarter Föderationskomponenten (Kapitel 4.5) und das Caching (Kapitel 5.5) bieten weitere Möglichkeiten zur Steigerung der Effizienz
Skalierbarkeit	• Basisarchitektur ermöglicht Duplizierung der Föderationskomponenten (Kapitel 3.1)

Die in dieser Arbeit vorgestellten Verfahren nutzen in erheblichem Maße die Charakteristika der Anwendungsdomäne der kontextbezogenen Anwendungen aus und erlauben auch insbesondere die Integration von Domänenwissen in den Anfrageverarbeitungsprozess. Die Eigenschaft typischer Anfragen, räumliche Prädikate und Typprädikate zu enthalten, wird im Verzeichnisdienst und in der orts- und typbezogenen Anfrageverarbeitung ausgenutzt. Die räumliche Beschränkung von Datensätzen (Dienstgebiet) wird bei der Verarbeitung von Gebietsanfragen wie auch Nachbarschaftsanfragen ausgenutzt. Der Ortsbezug der Daten wird beim Cachen der Daten in der Föderationskomponente, bei der Kooperation benachbarter Föderationskomponenten sowie bei der Speicherung von MRep-Relationenobjekten ausgenutzt. Die Eigenschaft des Datenmodells, Mehrfachattribute zu erlauben, erfordert besondere Berücksichtigung bei der formalen Definition der Anfragesemantik sowie bei der Behandlung von Mehrfachrepräsentationen. Da Geodaten häufig in verschiedenen Koordinatensystemen vorliegen wurde mit der automatischen Koordinatentransformation die Interoperabilität dieser Daten ermöglicht.

Ein Großteil der in dieser Arbeit vorgestellten Konzepte konnte bereits veröffentlicht werden. Die Basisarchitektur und das Gesamtkonzept des Nexus-Systems wurden beschrieben in [NGS+01, SHG+04, MNG+05]. [SIG+04] erläutert die föderierten Nachbarschaftsanfragen. [SHGN04] stellt die automatische Koordinatentransformation vor. Ideen zur Kooperation von Föderationskomponenten bei der Verwaltung von mobilen Objekten werden in [DSB+04] erarbeitet. In [GBH+05] wird beschrieben wie eine geeignete Implementierung je nach den Eigenschaften der bereitzustellenden Umgebungsmodelldaten ausgewählt werden kann. Die Erweiterung des Datenmodells um Attributteile, die insbesondere zur Repräsenta-

tion von Metadaten benötigt werden, wird in [HKNS05] beschrieben. Die orts- und typbezogene Anfrageverarbeitung wird in [SGNM06] erläutert.

Die hier vorgestellten Konzepte wurden überwiegend im Prototyp des Nexus-Systems umgesetzt so dass sie von Anwendungen und anderen Komponenten des Nexus-Systems genutzt werden können. Die Implementierung der Föderationskomponente konnte auf einer internationalen Konferenz [NGS03] sowie bei der Begehung des Sonderforschungsbereichs 627 im Juni 2006 gezeigt werden. Die Schnittstellensprachen der Implementierung werden in [BDG+04] definiert. Zahlreiche weitere Implementierungsaspekte sind in technischen Berichten dokumentiert. Diese Arbeit legt somit den Grundstein für weitere Forschungsarbeiten insbesondere auch der Projektpartner im Nexus-Projekt.

6.2 Ausblick

Neben den in den Kapiteln 4.5 (Kooperation benachbarter Föderationskomponenten) und 5.6 (Caching) vorgestellten weiteren Maßnahmen, welche auf die Steigerung der Effizienz bei der Erfüllung der bekannten Anforderungen abzielen, werden im Folgenden Fortentwicklungen der Integrationsmiddleware vorgestellt, welche im Zusammenhang mit zukünftigen, weitergehenden Anforderungen stehen.

Mächtigere Anfragesprache und Aktualisierungen

Es ist zu untersuchen, ob die bisherige einfache Anfragemächtigkeit auch für zukünftige komplexere kontextbezogene Anwendungen noch ausreichend ist. Räumliche Verbunde und Aggregationen stellen dabei die naheliegendsten Erweiterungen dar, weitere können mit einer Szenarienanalyse ermittelt werden. Mit räumlichen Verbunden können spezielle Kombinationen von Objekten erkannt werden, beispielsweise alle italienischen Restaurants entlang der Fahrtroute, bei denen es in weniger als 500 Metern Entfernung eine Tankstelle gibt. Mit einer Aggregation könnte die Durchschnittstemperatur in einem Gebäude oder die Gesamtzahl der freien Parkplätze rund um ein Einkaufszentrum ermittelt werden. Neben den algorithmischen Fragen, wie die Verbunde und Aggregationen berechnet werden und ob die Datenanbieter dafür zusätzliche Fähigkeiten besitzen müssen, muss auch geklärt werden, wie die Ergebnisse solcher Operatoren im Datenmodell des Umgebungsmodells repräsentiert werden können.

Bisher können Aktualisierungen nur unabhängig voneinander verarbeitet werden. Dies wird insbesondere dann zum Problem, wenn alle Repräsentationen eines mehrfach repräsentierten Objekts angepasst werden sollen. Wenn die Änderung von manchen Datenanbietern zurückgewiesen wird entsteht ein inkonsistenter Objektzustand. Dem kann begegnet werden, wenn Änderungen transaktional durchgeführt werden. Hierbei ist zu beachten, dass bei verteilten Transaktionen das zwei Phasen Commit Protokoll notwendig wird, wodurch die Autonomie der Datenanbieter eingeschränkt wird. Des Weiteren kann untersucht werden, wie ein generelles Transaktionskonzept umgesetzt werden kann, das mehrere Anfragen und Änderungen in einer atomaren Transaktion bündeln kann. Im Zusammenhang mit Caching müssen effiziente Verfahren zur Propagation solcher Änderungen ent-

wickelt werden. Das Caching Konzept kann ebenfalls hilfreich sein, wenn eine Transaktion zunächst nur auf der Cache-Kopie arbeitet und die Änderungen erst nach erfolgreichem Abschluss der Transaktion an den Datenanbieter übermittelt werden.

Operator-basierte Anfrageverarbeitung und Datenströme

Die bisherige Anfrageverarbeitung arbeitet nach einem festgelegten Ablaufschema, was für die bisherige Anfragemächtigkeit auch völlig ausreichend war. Nach einer Erweiterung der Anfragemächtigkeit muss auch darüber nachgedacht werden, wie die Anfrageverarbeitung flexibilisiert werden kann. Hierfür muss ein Operatorkonzept entwickelt werden, mit dem die Verarbeitung in einzelne Etappen mit klar definierter Schnittstelle aufgeteilt werden kann. Die bisher entwickelten Ergebnisaufbereitungsoperatoren sollten ebenfalls in dieses Operatorkonzept integriert werden können. Des Weiteren kann im Zuge der Kooperation benachbarter Föderationskomponenten untersucht werden, wie die Operatoren verteilt ausgeführt werden können, um die Last besser zu verteilen, und um den Weg der Daten zu optimieren, wenn die Operatoren näher zu den Daten verlagert werden können. Hierbei kann auf bekannten Konzepten aus den Bereichen der verteilten Anfrageverarbeitung [Koss00] und der Grid-basierten Anfrageverarbeitung [SWG+03] aufgebaut werden.

Je mehr Sensoren Daten für das Umgebungsmodell automatisch erheben, desto dynamischer wird das Umgebungsmodell selbst. Sensoren können dann einen Strom von Änderungen definieren anstatt ihre Daten über einzelne Aktualisierungsnachrichten in das Umgebungsmodell einzubringen. In gleicher Weise können Anwendungen auf dem aktuellen Stand bleiben, indem sie einen Datenstrom der Änderungen in einem bestimmten Ausschnitt des Umgebungsmodells anfordern anstatt immer wieder dieselbe Anfrage zu stellen. Hierbei kann untersucht werden, wie bisherige Datenstromverarbeitungsansätze [CHKS04, SBL04] in das Verarbeitungskonzept des Nexus-Systems integriert werden können, und ob die Verarbeitung von Datenströmen mit Operatoren beschrieben werden kann, die zu obigem Operatorkonzept kompatibel sind, so dass beide Verarbeitungsparadigmen kombiniert werden können. Zusätzlich kann darüber nachgedacht werden, ob auch der Event Service in dieses Operatorkonzept integriert werden kann. Hierbei können Erfahrungen aus anderen datenstromverarbeitenden Systemen [CLC+02, LWZ04], die ebenfalls ereignisbasierte Anfrageergebnisse integrieren, mit einbezogen werden.

Anfrageoptimierung

Nachdem die Mächtigkeit der Anfrageschnittstelle wesentlich erweitert worden ist und die Verarbeitung von Anfragen mittels des Operatorkonzept wesentlich flexibilisiert worden ist ergeben sich vielfältige neue Möglichkeiten zur Optimierung der Anfrageverarbeitung. Es können heuristische Transformationsregeln entwickelt werden, die immer zu einer Verbesserung führen. Für einige Operatoren können verschiedene Implementierungen zur Verfügung stehen, die mittels eines noch zu konzipierenden Kostenmodells gegeneinander abgewogen werden können. Darauf aufbauend kann ein kostenbasierter Optimierer entworfen werden, der die Reihenfolge der Operatoren im Anfragegraph optimiert. Dieser kann ebenfalls die Vertei-

lung der Operatoren auf die Föderationskomponenten und eventuell auch auf die Datenanbieter optimieren und dabei auch deren aktuellen Auslastungszustand berücksichtigen. Hierbei können Konzepte aus [LBR+05] mit einbezogen werden. Auf diese Weise kann durch die bessere Ausnutzung der verfügbaren Ressourcen die von einem Benutzer wahrgenommene Leistung des Gesamtsystems gesteigert werden ohne dass tatsächlich mehr Verarbeitungskapazität zur Verfügung gestellt werden muss.

7

Literaturverzeichnis

[AAH+97] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton: *A mobile contextaware tour guide*. Wireless Networks 3 (5), pp. 421-433, 1997

[ABK+03] M. Altinel, C. Bornhövd, S. Krishnamurthy, C. Mohan, H. Pirahesh, B. Reinwald: *Cache Tables: Paving the Way for an Adaptive Database Cache*. Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, September 9-12, 2003, Berlin, Germany, pp. 718-729, 2003

[ADD+05] B. Arbter, T. Drosdol, F. Dürr, M. Großmann, N. Hönle, S. Volz: *Das Nexus Relationen- und Topologiekonzept*, SFB 627 Bericht Nr. 2005/01, 2005

[APTP03] K. Amiri, S. Park, R. Tewari, S. Padmanabhan: *Scalable template-based query containment checking for web semantic caches*. Proceedings of the 19th International Conference on Data Engineering, ICDE 2003, Bangalore, India, March 5-8, 2003

[Arya95] S. Arya: *Nearest Neighbor Searching and Applications*, Ph.D. thesis, Dept. of Computer Science, University of Maryland, College Park, MD, USA, 1995

[AsCh75] M.M. Astrahan, D.D. Chamberlin: *Implementation of a Structured English Query Language*. Communications of the ACM, 18(10), pp. 580-588, 1975

[Attu01] Attunity: *Attunity Connect Architecture*. White paper, March 2001, http://www.attunity.com/files/AttunityConnectTechWhitePaper.pdf

[BaKi01] J. Barton, T. Kindberg: *The Cooltown User Experience*. Technical Report HPL-2001-22, Internet and Mobile Systems Laboratory, HP Laboratories Palo Alto, February 01, 2001, http://www.hpl.hp.com/techreports/2001/HPL-2001-22.html

[BaRo04] M. Bauer, K. Rothermel: *How to Observe Real-World Events through a Distributed World Model*. Proceedings of the Tenth International Conference on Parallel and Distributed Systems 2004 (ICPADS 2004); Newport Beach, California, July 7-9, 2004

[Barr03] D.K. Barry: *Web Services and Service-Oriented Architectures*. Morgan Kaufmann Publishers, 2003

[BBB+97] S. Berchtold, C. Böhm, B. Braunmüller, D. A. Keim, H.-P. Kriegel: *Fast Parallel Similarity Search in Multimedia Databases*, SIGMOD 1997, Proc. ACM SIG-

MOD International Conference on Management of Data, May 1997, Tucson, Arizona, USA, pp. 1-12

[BBHS03] M. Bauer, C. Becker, J. Hähner, and G. Schiele: ContextCube – Providing Context Information Ubiquitously, Proc. of the 23rd Intl. Conf. on Distributed Computing Systems Workshops (ICDCS 2003), 2003

[BDG+04] M. Bauer, F. Dürr, J. Geiger, M. Grossmann, N. Hönle, J. Joswig, D. Nicklas, T. Schwarz: *Information Management and Exchange in the Nexus Platform*. Version 2.0, Universität Stuttgart: Sonderforschungsbereich SFB 627, Fakultätsbericht Nr. 2004/04, 2004.

[Bea03] BEA Systems, Inc.: *Streaming API For XML*. JSR-173 Specification, Final v1.0, October 8th, 2003, http://www.jcp.org/en/jsr/detail?id=173

[BEK+98] S. Berchtold, B. Ertl, D. A. Keim, H.-P. Kriegel, T. Seidl: *Fast Nearest Neighbor Search in High-dimensional Space*, Proc. of the 14th Intl. Conf. on Data Engineering (ICDE'98), Orlando, Florida, Feb 1998

[Bern99] T. Berners-Lee: *Weaving the Web*. Harpur, San Francisco, 1999

[Bern02] L. Bernard: *Experiences from an implementation Testbed to set up a national SDI*. 5th AGILE Conf. on Geographic Information Science, Palma, Spain, 2002

[BGH+04] S. Bourbonnais, V.M. Gogate, L.M. Haas, R.W. Horman, S. Malaika, I. Narang, V. Raman: *Towards an information infrastructure for the grid*. IBM Systems Journal 43(4): 665-688, 2004

[BGK+02] P.A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, I. Zaihrayeu: *Data management for Peer-toPeer Computing: A Vision*. 5th Int'l. Workshop on the Web and Databases, Madison, Wisconsin, June 6-7, 2002

[BJS04] M. Bauer, L. Jendoubi, O. Siemoneit: *Smart Factory – Mobile Computing in Production Environments*. In: Proceedings of the MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004), 2004

[BKLW99] S. Busse, R.-D. Kutsche, U. Leser, H. Weber: *Federated Information Systems: Concepts, Terminology and Architectures*. Forschungsberichte des Fachbereichs Informatik Nr. 99-9, TU Berlin, April 1999

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger: *The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles,* Proc. of the 1990 ACM SIGMOD Intl. Conf. on Management of Data, Atlantic City, New Jersey, USA, ACM Press, 1990, pp. 322-331

[BLN86] C. Batini, M. Lenzerini, S.B. Navathe: *A Comparative Analysis of Methodologies for Database Schema Integration*. ACM Computing Surveys, Vol. 18, No. 4, Dec. 1986

[BlNa05] J. Bleiholder, F. Naumann: *Declarative Data Fusion – Syntax, Semantics, and Implementation*. Proceedings of the 9th East European Conference on Advances in Databases and Information Systems, ADBIS 2005, Tallinn, Estonia, September 12-15, 2005, pp. 58-73

[BMR04] S. Bürklen, P.J. Marron, K. Rothermel: *An Enhanced Hoarding Approach on Graph Analysis*. Proc. of the Intl. Conf. on Mobile Data Management (MDM), 2004

[Box00] D. Box et mult: *Simple Object Access Protocol (SOAP) 1.1.* W3C Note 08 May 2000, http://www.w3.org/TR/SOAP

[BQK96] P.A. Boncz, W. Quak, M.L. Kersten: *Monet And Its Geographic Extensions: A Novel Approach to High Performance GIS Processing*. Proc. of the 5th Intl. Conf. on Extending Database Technology (EDBT), Avignon, France, LNCS 1057, pp. 147-166, 1996

[BrPa98] S. Brin, L. Page: *The anatomy of a large-scale hypertextual Web search engine*. Proceedings of the Seventh International World Wide Web Conference, Computer Networks and ISDN Systems, Vol. 30, No. 1-7 , April 1998, pp. 107-117, 1998

[Care99] M. Carey et al.: *O-O, What Have They Done to DB2?* Proc. of 25th Intl. Conf. on Very Large Data Bases (VLDB), September 7-10, 1999

[CDK+02] Francisco Curbera, Matthew Duftler, Rania Khalaf,William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana: Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, Vol. 6, No. 2, Mar/Apr 2002, pp. 86-93, 2002

[CDM+00] K. Cheverst, N. Davies, K. Mitchell, A. Friday, C. Efstratiou: *Developing a Contextaware Electronic Tourist Guide: Some Issues and Experiences*. Proceedings of the Conference on Human Factors in Computing Systems CHI 2000, Netherlands, 2000

[CFFK01] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman: *Grid Information Services for Distributed Resource Sharing*. 10th IEEE International Symposium on High Performance Distributed Computing, 2001

[ChFu98] K. L. Cheung, A.W. Fu: *Enhanced Nearest Neighbour Search on the R-tree*, SIGMOD Record, vol. 27, no. 3, pp. 16-21, 1998

[CHKS04] M. Cammert, C. Heinz, J. Krämer, B. Seeger: *Anfrageverarbeitung auf Datenströmen*. Datenbank-Spektrum, 4. Jahrgang, Heft 11, pp. 5-13, November 2004

[CKW01] W.S. Conner, L. Krishnamurthy, R. Want: *Making Everyday Life Easier Using Dense Sensor Networks*. Proceedings of UBICOMP 2001, Atlanta, USA, 2001

[CLC+02] N. Cohen, H. Lei, P. Castro, J. Davis II, A. Purakayastha: *Composing Pervasive Data Using iQL*. 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002), 20-21 June 2002, Callicoon, NY, USA

[CNP99] P. Ciaccia, A. Nanni, M. Patella: *A Query-sensitive Cost Model for Similarity Queries with M-tree*, Proc. of the 10th Australasian Database Conference (ADC'99), Auckland, New Zealand, Jan 1999, pp. 65-76

[Codd86] E. F. Codd: *Missing Information (Applicable and Inapplicable) in Relational Databases*. SIGMOD Record 15(4): 53-78, 1986

[Codd87] E. F. Codd: *More Commentary on Missing Information in Relational Databases (Applicable and Inapplicable Information)*. SIGMOD Record 16(1): 42-50, 1987

[CoTe03] con terra GmbH: *terraCatalog*. http://www.conterra.de/

[CrBe97] M. E. Crovella, A. Bestavros: *Self-similarity in World Wide Web Traffic: Evidence and Possible Causes*. IEEE/ACM Trans. Netw., vol. 5, no. 6, pp. 835-846, 1997

[Davi98] J.R. Davis: *IBM's DB2 Spatial Extender: Managing Geo-Spatial Information Within The DBMS*. Technical Report, prepared for IBM Corporation, May, 1998

[DeAb99] A. Dey, G. Abowd: *Towards a better understanding of context and context-awareness*. Georgria Tech GVU Technical Report, GIT-GVU-99-22, 1999

[deeg04] deegree Project: *Building blocks for spatial data infrastructures*. deegree Java framework for geospatial solutions, founded by the GIS and Remote Sensing unit of the Department of Geography, University of Bonn, and lat/lon. http://deegree.sourceforge.net/

[Deel04] E. Deelman et al.: *Grid-Based Metadata Services*. Proc. of the 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM), Santorini Island, Greece, 2004

[Deng06] N. Dengler: *Entwurf und Implementierung eines Mehrwertdienstes zur Verarbeitung geometrischer 3D-Anfragen in Nexus*. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Diplomarbeit Nr. 2381, 2006

[DFJ+96] S. Dar, M.J. Franklin, B.T. Jónsson, D. Srivastava, M. Tan: *Semantic Data Caching and Replacement*. Proceedings of 22th International Conference on Very Large Data Bases, VLDB'96, Mumbai (Bombay), India, September 3-6, pp. 330-341, 1996

[DiFe01] Y. Ding, D. Fensel: *Ontology Library Systems: The Key to Successful Ontology Re-Use*. In Proc. 1st Int'l Semantic Web Working Symposium (SWWS'01), 2001

[DKL05] A. Dogac, Y. Kabak, G.B. Laleci: *Enhancing ebXML Registries to Make them OWL Aware*. Journal on Distributed and Parallel Databases, 18(1), pp. 9-36, July 2005

[DRS+98] P. Deshpande, K. Ramasamy, A. Shukla, J.F. Naughton: *Caching Multidimensional Queries Using Chunks*. Proceedings ACM SIGMOD International Conference on Management of Data, SIGMOD 1998, Seattle, Washington, USA, June 2-4, pp. 259-270, 1998

[DSB+04] T. Drosdol, T. Schwarz, M. Bauer, M. Großmann, N. Hönle, D. Nicklas: *Keeping Track of "Flying Elephants": Challenges in Large-Scale Management of Complex Mobile Objects*. Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), INFORMATIK 2004, Ulm, 20.-24. September 2004

[DüRo03] F. Dürr, K. Rothermel: *On a Location Model for Fine-Grained GeoCast*. 5th Intl. Conf. on Ubiquitous Computing, UbiComp03, 2003

[EgHe91] M.J. Egenhofer, J.R. Herring: *Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases*. Technical Report, Department of Surveying Engineering, University of Maine, Orono, ME, 1991

[Enge04] J. Enge: *Caching in der Nexus Föderation*. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Diplomarbeit Nr. 2179, 2004

[EPSG04] European Petroleum Survey Group (EPSG): *Geodesy Parameters V 6.6*. Released October 23, 2004, http://www.epsg.org/

[FAA99] H. Ferhatosmanoglu, D. Agrawal, A. El Abbadi: *Concentric Hyperspaces and Disk Allocation for Fast Parallel Range Searching*, Proceedings of the 15th

International Conference on Data Engineering, ICDE 1999, March 1999, Sydney, Austrialia, pp. 608-615

[FAA01] H. Ferhatosmanoglu, D. Agrawal, A. El Abbadi: *Optimal Partitioning for Efficient I/O in Spatial Databases*, Euro-Par 2001: Parallel Processing: 7th International Euro-Par Conference Manchester, UK August 28-31, 2001, LNCS 2150 / 2001, pp. 889-900

[FaBh93] C. Faloutsos, P. Bhagwat: *Declustering Using Fractals*, Proc. of the 2nd Intl. Conf. on Parallel and Distributed Information Systems (PDIS 1993), San Diego, CA, USA, January 1993, pp. 18-25

[Fagi96] R. Fagin: *Combining Fuzzy Information from Multiple Systems*, Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1996, June 3-5, 1996, Montreal, Canada, pp. 216-226

[FGDC] The Federal Geographic Data Committee: *The Clearinghouse*, http://www.fgdc.gov/clearinghouse/clearinghouse.html

[FHH+00] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, M. Klein: *OIL in a Nutshell*. Proc. of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management, Springer Verlag, 2000

[FoKe99] I. Foster, C. Kesselmann, eds.: *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufman, 1999

[GaGü98] V. Gaede, O. Günther: *Multidimensional Access Methods*. ACM Computing Surveys, 30(2), pp. 170-231, June 1998

[GBH+05] M. Grossmann, M. Bauer, N. Hönle, U.-P. Käppeler, D. Nicklas, T. Schwarz: *Efficiently Managing Context Information for Large-scale Scenarios*. 3rd IEEE Conf. on Pervasive Computing and Communications (PerCom'05), Kauai Island, Hawaii, March 8-12, 2005

[Gess90] G. H. Gessert: *Four Valued Logic for Relational Database Systems*. SIGMOD Record 19(1): 29-35, 1990

[GFSS00] H. Galhardas, D. Florescu, D. Shasha, E. Simon: *AJAX: An Extensible Data Cleaning Tool*. Demonstration, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD 2000, May 16-18, 2000, Dallas, Texas, USA, p.590

[GLM02] A. Gupta, B. Ludäscher, M. Martone: *Registering Scientific Information Sources for Semantic Mediation*. Proc. of the 21st Int'l. Conf. on Conceptual Modeling, Tampere, Finland, 2002

[GoLa01] J. Goldstein, P.-Å. Larson: *Optimizing Queries Using Materialized Views: A practical, scalable solution*. Proceedings of the ACM SIGMOD Conference on Management of Data, SIGMOD 2001, Santa Barbara, California, May 21-24, 2001

[Goog04] Google: *Google Local*. http://local.google.com/

[GöSe04] G. v. Gösseln, M. Sester: *Integration of Geoscientific Data Sets and the German Digital Map using a Matching Approach*. Proceedings of the XXth ISPRS Congress, Comm. IV, Istanbul, Turkey, 2004, pp. 1249-1254

[GPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, J. Widom: *The TSIMMIS Approach to Mediation: Data Models and Languages*. Journal of Intelligent Information Systems (JIIS) 8(2): 117-132, 1997

[GPZ01] S. Greco, L. Pontieri, E. Zumpano: *Integrating and Managing Conflicting Data*. Revised Papers from the 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics, PSI 2001, Akademgorodok, Novosibirsk, Russia, July 2-6, 2001, LNCS 2244, pp. 349-362

[GSW96] S. Guo, W. Sun, M.A. Weiss: *Solving Satisfiability and Implication Problems in Database Systems*. ACM Transactions on Database Systems (TODS), Vol. 21, No. 2, pp. 270-293, June 1996

[HäBü04a] T. Härder, A. Bühmann: *Query Processing in Constraint-Based Database Caches*. IEEE Data Engineering Bulletin, Vol. 27, No. 2, pp. 3-10, March 2004

[HäBü04b] T. Härder, A. Bühmann: *Datenbank-Caching - Eine systematische Analyse möglicher Verfahren*. Informatik - Forschung und Entwicklung, Band 19, Heft 1, pp. 2-16, Juli 2004

[Hale01] A.Y. Halevy: *Answering queries using views: A survey*. VLDB Journal 10(4), pp. 270-294, 2001.

[HäRa01] T. Härder, E. Rahm: *Datenbanksysteme: Konzepte und Techniken der Implementierung*, 2. Auflage Springer 2001

[Haun05] J.-H. Haunert: *Link based Conflation of Geographic Datasets*. Proceedings of the ICA workshop on Generalisation and Multiple Representation, A Coruna, Spain, 2005

[HDT04] A. Hub, J. Diepstraten, T. Ertl: *Design and Development of an Indoor Navigation and Object Identification System for the Blind*. Proceedings of the 6th ACM SIGACCESS Conference on Computers and Accessibility ASSETS 2004, October 18-20, Atlanta, GA, USA, pp. 147-152, 2004

[HeGu00] J. Hendler, D. McGuinness: *The DARPA Agent Markup Language*. IEEE Intelligent Systems, Vol. 15, No. 6, 2000

[HIST03] A.Y. Halevy, Z.G. Ives, D. Suciu, I. Tatarinov: *Schema Mediation in Peer Data management Systems*. 19th Int'l. Conf. on Data Engineering, Bangalore, India, March 5-8, 2003

[HjSa95] G. R. Hjaltason, H. Samet: *Ranking in Spatial Databases*, Proc. of the 4th Symp. on Spatial Databases, Portland, Maine, USA, Aug. 1995, LNCS 951, pp. 83-95

[HjSa99] G. R. Hjaltason, H. Samet: *Distance Browsing in Spatial Databases*, ACM Transactions on Database Systems, vol. 24, no. 2, pp. 265-318, 1999

[HJK+01] W. Hoschek, J. Jaen-Martinez, P. Kunszt, B. Segal, H. Stockinger, K. Stockinger, B. Tierney: *Data Management Architecture Report. Design, Requirements and Evaluation Criteria*. Technical report, DataGrid-02-D2.2, 2001.

[HKL+99] F. Hohl, Uwe Kubach, A. Leonhardi, K. Rothermel, M. Schwehm: *Next Century Challenges: Nexus—An Open Global Infrastructure for Spatial-Aware Applications*. Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom '99) Seattle, WA, USA, August 1999

[HKNS05] N. Hönle, U.-P. Käppeler, D. Nicklas, T. Schwarz: *Benefits Of Integrating Meta Data Into A Context Model*. Proc. 2nd IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea), @ 3rd IEEE Intl. Conf. on Pervasive Computing and Communication (PerCom'05), Hawaii, March 12, 2005

[HMN+99] L.M. Haas, R.J. Miller, B. Niswonger, M. Tork Roth, P.M. Schwarz, E.L. Wimmers: *Transforming Heterogeneous Data with Database Middleware: Beyond Integration*. IEEE Data Engineering Bulletin, Vol. 22, No. 1, pp. 31-36, 1999

[Hosc02] W. Hoschek: *A Database for Dynamic Distributed Content and its Application for Service and Resource Discovery*. In Intl. IEEE Symp.on Parallel and Distributed Computing (ISPDC 2002), Iasi, Romania, July 2002

[HSS04] H. Höpfner, S. Schosser, K.-U. Sattler: *An Indexing Scheme for Update Notification in Large Mobile Information Systems*. Proceedings of the Workshop on Pervasive Information Management @ 9th International Conference on Extending Database Technology, EDBT Workshops 2004, Heraklion, Crete, Greece, March 18, pp. 345-354, 2004

[HXW+05] H. Hu, J. Xu, W.S. Wong, B. Zheng, D.L. Lee, W.-C. Lee: *Proactive Caching for Spatial Queries in Mobile Environments*. Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, Tokyo, Japan, 5-8 April, pp. 403-414, 2005

[IBM00] IBM Corporation: *IBM DB2 Data Joiner*. Fact sheet, 2000, http://www-3.ibm.com/software/data/datajoiner/brochure/factsheet.pdf

[ImLi84] T. Imielinski, W. Lipski: *Incomplete Information in relational databases*. Journal of the ACM 31(4): 761-791, October, 1984

[INV91] T. Imielinski, S. A. Naqvi, K. V. Vadaparty: *Incomplete Objects – A Data Model for Design and Planning Applications*. Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data, SIGMOD 1991, Denver, Colorado, May 29-31, 1991, pp. 288-297

[ISO99] ISO: *Information technology - Database languages - SQL multimedia and application packages - Part 3: Spatial*. ISO/IEC 13249-3:1999, 1999

[KaFe06] M. Kada, L. Fen: *Generalisation of Building Ground Plans using Half Spaces*. In: Proceedings of the International Symposium on Geospatial Databases for Sustainable Development, Goa, India, September 27-30, 2006

[KeBa96] A.M. Keller, J. Basu: *A Predicate-based Caching Scheme for Client-Server Database Architectures*. The VLDB Journal, Vol. 5 No. 1, pp. 35-47, January 1996

[KFK96] N. Koudas, C. Faloutsos, I. Kamel: *Declustering Spatial Databases on a Multi-Computer Architecture*, Proc. of the 5th International Conference on Extending Database Technology, EDBT'96, Avignon, France, March 1996, pp. 592-614

[KGP05] T.H. Kolbe, G. Gröger, L. Plümer: *CityGML – Interoperable Access to 3D City Models*. Proc. of the 1st Intl. Symp. on Geo-Information for Disaster Management (GI4DM), Delft, The Netherlands, Springer, 2005

[KKD89] W. Kim, K.C. Kim, A. Dale: *Indexing techniques for object-oriented databases*. In: W. Kim, F.H. Lochovsky (eds.): "Object-Oriented Concepts, Databases, and Applications", Addison-Wesley, pp. 371-394, 1989

[KKN+05] U.-P. Käppeler, G. Kindermann, D. Nicklas, N. Hönle, D. Dudkowski: *Shared Dynamic Context Models: Benefits for Advanced Sensor Data Fusion for Autonomous Robots*. Proc. Artificial Intelligence and Applications 2005, Innsbruck, Austria, February 14-16, 2005

[KOA+99] C. Kidd, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner, W. Newstetter: *The Aware Home: A Living Laboratory for Ubiquitous Computing Research*. Proceedings of 2nd International Workshop on Cooperative Buildings—CoBuild, 1999

[Koss00] D. Kossmann: *The State of the art in distributed query processing*. ACM Computing Surveys, Vol. 32, No. 4, pp. 422-469, December 2000

[Kund06] S. Kundu: *Conflating two Polygonal Lines*. Pattern Recognition 39(3), 2006, pp. 363-372

[LaYa85] Per-Åke Larson, H.Z. Yang: *Computing Queries from Derived Relations*. Proc. of the 11th International Conference on Very Large Data Bases, VLDB 1985, Stockholm, Sweden, pp. 259-269, August 21-23, 1985

[LBBN04] O. Lehmann, M. Bauer, C. Becker, D. Nicklas: *From Home to World - Supporting Context-aware Applications through World Models*. Proc. Second IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom'04), 2004

[LBR+05] W.-S. Li, V.S. Batra, V. Raman, W. Han, K.S. Candan, I. Narang: *Load and Network Aware Query Routing for Information Integration*. Proc. 21st Intl. Conf. on Data Engineering, ICDE 2005, Tokyo, Japan, 5-8 April 2005

[LeRo02] A. Leonhardi, K. Rothermel: *Architecture of a largescale location service*. 22nd Intl. Conf. on Distributed Computing Systems (ICDCS), 2002

[LHY97] Y.-l. Lo, K. A. Hua, H. C. Young: *A General Multidimensional Data Allocation Method for Multicomputer Database Systems*, Database and Expert Systems Applications, 8th International Conference, DEXA '97, Toulouse, France, September 1-5, 1997, pp. 357-366

[LLN02] Dan-Zhou Liu, Ee-Peng Lim, Wee-Keong Ng: *Efficient k Nearest Neighbor Queries on Remote Spatial Databases Using Range Estimation*, Proc. of the 14th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'02), Edinburgh, Scotland, July 2002, pp. 121-130

[LLS99] K.C.K. Lee, H.V. Leong, A. Si: *Semantic query caching in a mobile environment*. Mobile Computing and Communications Review, Vol. 3, No. 2, pp. 28-36, April 1999

[LLS02] K.C.K. Lee, H.V. Leong, A. Si: *Semantic Data Broadcast for a Mobile Environment Based on Dynamic and Adaptive Chunking*. IEEE Transactions on Computers, Vol. 51, No. 10, pp. 1253-1268, October 2002

[LMS05] P. Leach, M. Mealling, R. Salz: A Universally Unique IDentifier (UUID) URN Namespace. IETF RFC 4122, July 2005, http://www.ietf.org/rfc/rfc4122.txt

[LOL92] C.C. Low, B.C. Ooi, H. Lu: *H-trees: A Dynamic Associative Search Index for OODB*. Proc. of the 1992 ACM SIGMOD Intl. Conf. on Manangement of Data, San Diego, California, pp. 134-143, 1992

[LRIE01] Y.G. Leclerc, M. Reddy, L. Iverson, M. Eriksen: *The GeoWeb - A New Paradigm for Finding Data on the Web*. 20th International Cartographic Conference, ICC, Beijing, China, 2001

[LRO96] A.Y. Levy, A. Rajaraman, J.J. Ordille: *Querying Heterogeneous Information Sources Using Source Descriptions*. Proceedings of 22th International Conference on Very Large Data Bases, VLDB'96, September 3-6, 1996

[LWZ04] Y. Law, H. Wang, C. Zaniolo: *Query Languages and Data Models for Database Sequences and Data Streams*. Proc. 30th Intl. Conf. on Very Large Data Bases (VLDB'04), Toronto, Canada, August 31 - September 3 2004

[MBK00] S. Manegold, P.A. Boncz, M.L. Kersten: *Optimizing database architecture for the new bottleneck: memory access*. VLDB Journal, 9(3), pp. 231-246, Dec 2000

[Mess01] J. Messmer: *Modellierung der Augmented World in Nexus*. Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1870, 2001

[MHH+01] R.J. Miller, M.A. Hernández, L.M. Haas, L. Yan, C.T.H. Ho, R. Fagin, L. Pope: *The Clio Project: Managing Heterogeneity*. SIGMOD Rec. Vol. 30 Nr. 1, ACM Press, 2001

[MNG+05] B. Mitschang, D. Nicklas, M. Großmann, T. Schwarz, N. Hönle: *Federating Location-Based Data Services*. In T. Härder, W. Lehner (Eds.): Data Management in a Connected World, Essays Dedicated to Hartmut Wedekind on the Occasion of His 70th Birthday, LNCS 3551, Springer, 2005

[MoAn04] A. Motro, P. Anokhin: *Fusionplex: Resolution of Data Inconsistencies in the Integration of Heterogeneous Information Sources*. Information Fusion, Vol. 7, No. 2, June 2006, pp. 176-196

[Morr90] J. M. Morrissey: *Imprecise Information and Uncertainty in Information Systems*. ACM Transactions on Information Systems (TOIS) 8(2): 159-180, 1990

[MuPo97] T.A. Mueck, M.L. Polaschek: *A configurable type hierarchy index for OODB*. VLDB Journal, 6(4), pp. 312-332, 1997

[MSS+02] A. Maedche, S. Staab, R. Studer, Y. Sure, R. Volz: *SEAL—Tying up Information Integration and Web Site Management by Ontologies*. Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 25, Nr. 1, March 2002

[Nebe96] D. Nebert: *Information Architecture of a Clearinghouse*. WWW Conference, 1996, http://www.fgdc.gov/publications/documents/clearinghouse/clearinghouse1.html

[Neum06] C. Neumann: *Graphische Repräsentation ortsbezogener Anfragesprachen*. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Diplomarbeit, 2006

[NGS+01] D. Nicklas, M. Großmann, T. Schwarz, S. Volz, B. Mitschang: *A Model-Based, Open Architecture for Mobile, Spatially Aware Applications*, Proc. of the 7th

Intl. Symp. on Spatial and Temporal Databases (SSTD01), Los Angeles, LNCS 2121, 2001, pp. 117-135

[NGS03] D. Nicklas, M. Grossmann, T. Schwarz: *NexusScout: An Advanced Location-Based Application On A Distributed, Open Mediation Platform*. Proceedings of the 29th VLDB Conference (Demonstration track), Berlin, Germany, 2003

[NHMM04] D. Nicklas, N. Hönle, M. Moltenbrey, B. Mitschang: *Design and Implementation Issues for Explorative Location-based Applications: the NexusRallye*. VI Brazilian Symposium on GeoInformatics (GeoInfo), November 22-24, 2004

[Nick00] D. Nicklas et. al.: *Final Report of Design Workshop*. Universität Stuttgart: Center of Excellence 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme), Fakultätsbericht Nr. 2000/08, 2000

[Nick05] D. Nicklas: *Ein umfassendes Umgebungsmodell als Integrationsstrategie für ortsbezogene Daten und Dienste*. Dissertation, Universität Stuttgart, 2005

[NiMi04] D. Nicklas, B. Mitschang: *On building location aware applications using an open platform based on the NEXUS Augmented World Model*. Software and Systems Modeling, Vol. 3(4), 2004

[Nove01] Novell, Inc.: *Which directory offers the best LDAP server?* White Paper, 2001, http://www.novell.com/info/collateral/docs/4621218.01/4621218.pdf

[Oasis04] Organization for the Advancement of Structured Information Standards (OASIS): *Introduction to UDDI: Important Features and Functional Concepts*. UDDI Technical White Paper, http://www.uddi.org/whitepapers.html, October 2004

[OGC99] Open GIS Consortium (OGC): *OpenGIS Simple Features Specification for CORBA*. Revision 1.1, June 2, 1999, http://www.opengeospatial.org/standards/sfc

[OGC99a] Open GIS Consortium (OGC): *OpenGIS Simple Features Specification for SQL*. Revision 1.1, Project Document 99-049, May 1999, http://www.opengis.org/docs/99-049.pdf

[OGC99b] OpenGIS Consortium (OGC): *Topic 8: Relationships between Features*. OpenGIS Abstract Specification, Version 4, Project Document Number 99-108r2, March 1999, http://portal.opengeospatial.org/files/?artifact_id=894

[OGC99c] Open GIS Consortium (OGC): *Topic 13: Catalog Services*, OpenGIS Abstract Specification, Version 4, Project Document Number 99-113, March 1999, http://portal.opengeospatial.org/files/?artifact_id=901

[OGC02] Open GIS Consortium (OGC): *OpenGIS Catalog Services Specification*, OGC Implementation Specification, Version 1.1.1, Project Document 02-087r3, Dec. 2002, http://portal.opengeospatial.org/files/?artifact_id=3843

[OGC03] Open GIS Consortium (OGC): *OWS1.2 UDDI Experiment*, OpenGIS Interoperability Program Report, Version 0.5, Jan 17, 2003, http://portal.opengeospatial.org/files/?artifact_id=1317

[OGC05a] Open Geospatial Consortium (OGC): *OGC Catalogue Services Specification*. OGC Implementation Specification, Version 2.0.1, Project Document 04-021r3, May 20, 2005, http://portal.opengeospatial.org/files/?artifact_id=5929&version=2

[OGC05b] Open Geospatial Consortium (OGC): *Web Feature Service Implementation Specification*. OpenGIS® Implementation Specification, Version 1.1, Project Document 04-094, May 3, 2005, https://portal.opengeospatial.org/files/?artifact_id=8339

[OMG00] Object Management Group, Inc.: *Trading Object Service Specification*. Version 1.0, May 2000, http://www.omg.org/cgi-bin/doc?formal/2000-06-27

[OMG05] Object Management Group, Inc.: *Meta Object Facility (MOF) Specification*. Version 1.4.1, ISO/IEC 19502:2005(E), July 2005

[PAAA98] S. Prabhakar, K. A. S. Abdel-Ghaffar, D. Agrawal, A. El Abbadi: *Cyclic Allocation of Two-Dimensional Data*, Proc. of the 14th Intl. Conf. on Data Engineering, ICDE 1998, February 1998, Orlando, Florida, USA, pp. 94-101

[PAG96] Y. Papakonstantinou, S. Abiteboul, H. Garcia-Molina: *Object Fusion in Mediator Systems*. Proceedings of 22th International Conference on Very Large Data Bases, VLDB 1996, September 3-6, 1996, Mumbai (Bombay), India, pp. 413-424

[PaMa96] A. Papadopoulos, Y. Manolopoulos: *Parallel Processing of Nearest Neighbor Queries in Declustered Spatial Data*, Proc. of the fourth ACM workshop on Advances in Geographic Information Systems, ACM-GIS 1996, Rockville, Maryland, United States, pp. 35-43

[PaMa98] A. Papadopoulos, Y. Manolopoulos: *Similarity Query Processing Using Disk Arrays*, SIGMOD 1998, Proc. ACM SIGMOD Intl. Conf. on Management of Data, June 1998, Seattle, Washington, USA, pp. 225-236

[PaMa01] A. Papadopoulos, Y. Manolopoulos: *Distributed Processing of Similarity Queries*, Distributed and Parallel Databases, Volume 9, Issue 1, January 2001, pp. 67 - 92

[PaVa02] Y. Papakonstantinou, V. Vassalos: *Achitecture and Implementation of an XQuery-based Information Integration Platform*. Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society, Vol. 25, Nr. 1, March 2002

[PBJ03] H. Pinto, N. V. Boas, R. José, *Using a private UDDI for publishing location-based information to mobile users*, ICCC/IFIP 7th Intl. Conf. on Electronic Publishing (ElPub2003), Guimarães, Portugal, June 2003

[RaHe01] V. Raman, J. M. Hellerstein: *Potter's Wheel: An Interactive Data Cleaning System*. Proceedings of 27th International Conference on Very Large Data Bases, VLDB 2001, September 11-14, 2001, Roma, Italy, pp. 381-390

[RaKa95] S. Ramaswamy, P.C. Kanellakis: *OODB Indexing by Class-Division*. Proc. of the 1995 ACM SIGMOD Intl. Conf. on Manangement of Data, San Jose, CA, pp. 139-150, 1995

[RDK03] Qun Ren, Margaret H. Dunham, Vijay Kumar: *Semantic Caching and Query Processing*. IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No.1, pp. 192-210, Jan/Feb 2003

[ReDu00] Q. Ren, M.H. Dunham: *Using semantic caching to manage location dependent data in mobile computing*. Proceedings of the sixth ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM 2000, Boston, MA, USA, pp. 210-221, August 6-11, 2000

[RKV95] N. Roussopoulos, S. Kelley, F. Vincent: *Nearest Neighbor Queries*, Proc. of the 1995 ACM-SIGMOD Intl. Conf. on Management of Data, San Jose, CA, USA, May 1995, pp. 71-79

[RNC+02] V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, C. Baru: *Data Access and Management Services on Grid*. 5th Global Grid Forum (GGF5) Workshop, 2002, http://www.gridforum.org/Meetings/GGF5/papers.htm

[RoHu80] D.J. Rosenkrantz, H.B. Hunt III: *Processing Conjunctive Predicates and Queries*. Sixth International Conference on Very Large Data Bases, VLDB 1980, Montreal, Quebec, Canada, pp. 64-72, October 1-3, 1980

[Ross04] K.A. Ross: *Selection Conditions in Main Memory*. ACM Trans. on Database Systems, 29(1), pp. 132-161, March 2004

[Roub03] V. Roubtsov: *My kingdom for a good timer! Reach submillisecond timing precision in Java*. JavaWorld, Java Q&A, January 10, 2003, http://www.javaworld.com/javaworld/javaqa/2003-01/01-qa-0110-timing.html

[RWM02] A. Rajasekar, M. Wan, R. Moore: *MySRB & SRB - Components of a Data Grid*. In International Symposium on High Performance Distributed Computing, 2002

[Saal87] A. Saalfeld: *Conflation: Automated Map Compilation*. Bureau of the Census, Statistical Research Division report series, report number: Census/SRD/RR-87/24, 1987

[SAB+95] V. S. Subrahmanian, S. Adah, A. Brink, R. Emery, A. Rajput, R. Ross, T. Rogers, C. Ward: *HERMES: A Heterogeneous Reasoning and Mediator System*. Technical Report, University of Maryland, 1995, www.cs.umd.edu/projects/hermes/ overview/paper

[SAC+79] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, T.G. Price: *Access Path Selection in a Relational Database Management System*. Proc. of the 1979 ACM SIGMOD Intl. Conf. on Manangement of Data, Boston, Massachusetts, 1979

[SAD+06] M. Scharf, B. Arbter, F. Dürr, T. Schwarz, O. Simoneit: *Performance Aspects of Large-scale Location-based Services*. Universität Stuttgart, SFB 627 Bericht SFB627-2006-08, 2006

[Same90] H. Samet: *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990

[SBL04] S. Schmidt, H. Berthold, W. Lehner: *QStream: Deterministic Querying of Data Streams*. Proc. 30th Intl. Conf. on Very Large Data Bases (VLDB'04), Toronto, Canada, August 31 - September 3 2004

[Scha05] M. Scharf: *On the Response Time of Large-scale Composite Web Services*. Proc. of the 19th International Teletraffic Congress (ITC 19), Beijing, 2005

[SCS00] K.-U. Sattler, S. Conrad, G. Saake: *Adding Conflict Resolution Features to a Query Language for Database Federations*. Proceedings of the 3rd Workshop on Engineering Federated Information Systems, EFIS 2000, June 19-20, 2000, Dublin, Ireland, pp. 41-52

[SDA99] D. Salber, A. Dey, G. Abowd: *The Context Toolkit: Aiding the Development of Context-Enabled Applications*. Proceedings of the Conference on Human Factors in Computing Systems CHI, 1999

[SGNM06] T. Schwarz, M. Grossmann, D. Nicklas, B. Mitschang: *Exploiting Type and Space in a Main Memory Query Engine*. Proceedings for the VIII Brazilian Symposium on GeoInformatics: GeoInfo 2006; Campos do Jordão, Brasil, November 19-22, 2006

[SHG+04] T. Schwarz, N. Hönle, M. Großmann, D. Nicklas, B. Mitschang: *Efficient Domain-Specific Information Integration in Nexus*. Proc. of the 2004 VLDB Workshop on Information Integration on the Web (IIWeb04), Web, @ 30th Intl. Conf. on Very Large Data Bases (VLDB'04), Toronto, Canada, September 4, 2004

[SHGN04] T. Schwarz, N. Hönle, M. Großmann, D. Nicklas: *A Library for Managing Spatial Context Using Arbitrary Coordinate Systems*. Proc. 1st IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea), @ 2nd IEEE Intl. Conf. on Pervasive Computing and Communication (PerCom'04), 2004

[SIG+04] T. Schwarz, M. Iofcea, M. Großmann, N. Hönle, D. Nicklas, B. Mitschang: *On Efficiently Processing Nearest Neighbor Queries in a Loosely Coupled Set of Data Sources*. Proc. 12th ACM Intl. Symp. on Advances in Geographic Information Systems (ACM GIS 2004), Washington D.C., November 12-13, 2004

[Sing03] G. Singh et al.: *A Metadata Catalog Service for Data Intensive Applications*. Proc. of the 2003 ACM/IEEE Conf. on Supercomputing (SC), Phoenix, Arizona, 2003

[STM00] A. Schmidt, A. Takaluoma, J. Mäntyjärvi: *Context-aware telephony over WAP*. Personal Technologies 4 (4), pp. 225-229, 2000

[Sun04] Sun Microsystems, Inc.: *The Collections Framework*. JDK 5.0 Documentation, http://java.sun.com/j2se/1.5.0/docs/guide/collections/index.html

[SWG+03] J. Smith, P. Watson, A. Gounaris, N.W. Paton, A.A.A. Fernandes, R. Sakellariou: *Distributed Query Processing on the Grid*. Intl. Journal of High Performance Computing Applications 17(4): 353-367, November 2003

[SWMY00] W. Smith, A. Waheed, D. Meyers, J. Yan: *An Evaluation of Alternative Designs for a Grid Information Service*. 9th IEEE International Symposium on High Performance Distributed Computing, 2000

[UCB02] U.S. Census Bureau: *TIGER/Line Files Technical Documentation*. April 2002, http://www.census.gov/geo/www/tiger/tigerua/ua2ktgr.pdf

[UCB03] U.S. Census Bureau: *TIGER/Line Files*. http://www.census.gov/geo/www/tiger/

[UDDI00] UDDI: *The UDDI Technical White Paper*, UDDI.org, Sept. 2000, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[VBD+01] J. Van den Bercken, B. Blohsfeld, J.-P. Dittrich, J. Krämer, T. Schäfer, M. Schneider, B. Seeger: *XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries*. Proc. of 27th Intl. Conf. on Very Large Data Bases (VLDB), Roma, Italy, pp. 39-48, 2001

[ViSo04] Vivid Solutions: *JTS Topology Suite*. http://www.vividsolutions.com/jts/jts-home.htm

[VJJS05] S. Vaid, C.B. Jones, H. Joho, M. Sanderson: *Spatio-textual Indexing for Geographical Search on the Web*. Proc. of the 9th Intl. Symp. on Spatial and Temporal Databases (SSTD), Angra dos Reis, Brazil, LNCS 3633, pp. 218-235, 2005

[VKM03] U. Vallamsetty, K. Kant, P. Mohapatra: *Characterization of E-commerce Traffic*. Electronic Commerce Research, vol. 3, no. 1-2, pp. 167-192, 2003

[Volz06a] S. Volz: *An Iterative Approach for Matching Multiple Representations of Street Data*. Proccedings of the JOINT ISPRS Workshop on Multiple Representations and Interoperability of Spatial Data, Hannover, Germany, February 22-24, 2006, pp. 101-110

[Volz06b] S. Volz: *Management and Conflation of Multiple Representations within an Open Federation Platform*. In: J.-R. Sack, M. Sester, P. van Oosterom, M. Worboys (eds.): Spatial Data: mining, processing and communicating, Dagstuhl Seminar Proceedings No. 06101, 2006, http://drops.dagstuhl.de/opus/volltexte/2006/588

[VoWa04] S. Volz, V. Walter: *Linking Different Geospatial Databases by Explicit Relations*. Proceedings of the XXth ISPRS Congress, Comm. IV, Istanbul, Türkei, 2004, pp. 152-157

[Vrho04] M. Vrhovnik: *Update-Propagation in gestaffelten und verteilten Caches*. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Diplomarbeit Nr. 2229, 2004

[W3C02] W3C: *OWL Web Ontology Language 1.0 Reference*. W3C Working Draft, 2002, http://www.w3.org/TR/2002/WD-owl-ref-20020729/

[WaFr99] V. Walter, D. Fritsch: *Matching Spatial Data Sets: a Statistical Approach*. Int. J. Geographical Information Science 13(5), 1999, pp. 445-473

[Walt97] V. Walter: *Zuordnung von raumbezogenen Daten – am Beispiel der Datenmodelle ATKIS und GDF*. Dissertation, Deutsche Geodätische Kommission (DGK), Reihe C, Heft Nr. 480, 1997

[WaZa00] H. Wang, C. Zaniolo: *Using SQL to Build New Aggregates and Extenders for Object- Relational Systems*. Proceedings of 26th International Conference on Very Large Data Bases, VLDB 2000, September 10-14, 2000, Cairo, Egypt, pp. 166-175

[Weis91] M. Weiser: *The Computer for the Twenty-First Century*. Scientific American, pp. 94-100, September 1991

[XiSp04] D. Xiong, J. Sperling: *Semiautomated Matching for Network Database Integration*. ISPRS Journal of Photogrammetry and Remote Sensing, Special Issue on

Advanced Techniques for Analysis of Geo-spatial Data, Volume 59, No. 1-2, 2004, pp. 35-46

[YaÖz99] L.-L. Yan, M. T. Özsu: *Conflict Tolerant Queries in AURORA*. Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems, CoopIS 1999, Edinburgh, Scotland, September 2-4, 1999, pp. 279-290

[YeMa06] YellowMap AG: *YellowMap - Alles in Ihrer Nähe*. http://www.yellowmap.de

[YOTJ01] C. Yu, B.C. Ooi, K.-L. Tan, H.V. Jagadish: *Indexing the Distance: An Efficient Method to KNN Processing*, Proc. of the 27th VLDB Conf., Roma, Italy, 2001

[Zhao04] P. Zhao et al.: *Grid Metadata Catalog Service-Based OGC Web Registry Service*. Proc. of the 12th annual ACM Intl. workshop on Geographic information systems (ACM GIS), Washington DC, 2004