

# The DYNAQUEST-Framework for Dynamic and Adaptive Source Selection

Marco Grawunder, Frank Köster

University of Oldenburg, Department of Computer Sciences  
Escherweg 2, 26125 Oldenburg, Germany,

Email: {grawunder|koester}@informatik.uni-oldenburg.de

## Abstract

Information overload is one of the main reasons why efficient use of internet information sources is a mostly cumbersome process. Closely connected to this general observations is the problem of resource discovery: Which are the best information sources to answer my query respecting my preferences in the current execution context (e.g. time of day).

*Virtual databases* try to free users from the burden of distribution and heterogeneity regarding the data access and data quality of information source in the Internet. Queries are formulated against one (or more) virtual views, relevant sources are determined by the system, queried, and finally the results of these sources are combined.

A latent problem in open systems like the Internet is caused by their high dynamics: A previously fast answering server can be unreachable or new and better-suited sources may be available. To cope with such problems, intelligent and fault-tolerant mechanisms are needed.

DYNAQUEST, a framework for dynamic and adaptive query processing, is our proposal to build robust virtual database systems. This agent-based approach helps locating relevant sources based on user preferences concerning data and quality aspects. While processing a query with these selected sources, errors (e.g. a source failure) are caught and DYNAQUEST tries to recover the query instead of breaking it (e.g. by choosing an alternate source). Our approach always tries to comply with desired information quality demands while optimizing or adapting query processing.

**Keywords:** agent-based, mediator architecture, source description, dynamic query processing, adaptive query processing

## 1 INTRODUCTION AND OVERVIEW

Internet technology enables access to distributed and heterogeneous data sources that provide valuable information for many applications. Nevertheless, users who want to operate in an open system like the Internet have to deal with problems concerning distribution, heterogeneity, dynamics, huge data volumes, and uncertain data quality.

To gainfully utilize the Internet as an information resource, relevant sources have to be selected, information from these sources have to be extracted, integrated, and application specific prepared and visualized. Nowadays, these tasks are mostly in charge of the user. Even worse, possible source failures in the high dynamic Internet must be handled manually. According to these observations, one can easily see that a better user assistance is desirable for all of these tasks.

The acceptance of such assistance systems depends highly on quality assessments of the gained results, the time to wait for results, and the necessary user interactions. Some important aspects in this context are the quality assessments of gained results, time to wait for these results, and the complexity of necessary user interactions within the context of query processing.

DYNAQUEST is our approach to cope with the mentioned problems. DYNAQUEST is an agent-based framework for dynamic and adaptive query processing which initially will be evaluated in typical scenarios of Internet-based electronic commerce (e.g. car trade). This framework focuses mainly on query processing, but observes also the above-sketched requirements regarding quality assurance and usability.

This paper is organized as follows. In Section 2 we provide background information concerning different aspects of our framework and compare our work to other approaches. Section 3 introduces DYNAQUEST's architecture (Sect. 3.1), followed by a specification of our source description framework (SDF) (Sect. 3.2). Section 3.3 deals with the source se-

lection process based on the SDF and multi-criteria analysis. Section 4 concludes the paper and gives an outlook on future work.

## 2 DYNAMIC AND ADAPTIVE QUERY PROCESSING

In our work we use the term *virtual database system (VirtDBS)*<sup>1</sup> to describe a database system which follows the mediator approach [36] (cf. Fig. 1). Although, the autonomous and heterogeneous sources which form this virtual database are distributed over the Internet, the user needs no knowledge about this distribution — he can use a VirtDBS like a standard database system. Normally, the distributed, heterogeneous and autonomous sources are wrapped<sup>2</sup> (e.g. [2, 3, 19]) and integrated through distributed query processing techniques [24]. Due to the source characteristic such systems are always read-only.

In this work, we extend the definition of virtual databases to systems, which dynamically integrates heterogeneous distributed sources [13]. The integration process takes into account the current execution context, i.e. the query itself, the user preferences concerning the query, and general quality expectations, as well as the current environment (e.g. time of day, server loads etc.). In contrast to other integration systems like federated database systems [31] or data warehouse systems [22] the source selection cannot be done statically at implementation time (*in advance integration*) but has to occur dynamically during query processing (*on demand integration*) to reflect the current processing context.

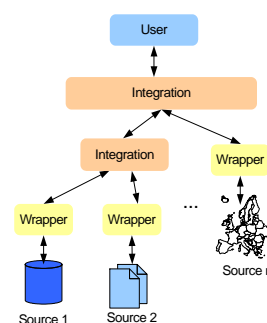


Figure 1: Architecture of a Virtual Database System

One of the main tasks the VirtDBS, we consider, has to solve, is the location of sources that could be *relevant* to a specific query. In this context, a source is relevant if it provides a positive contribution to the query solution [32]. The relevance can be seen as the distance between an optimal solution and the proposed solution and concerns different aspects:

<sup>1</sup>Originally this term was defined in [16], but we use the term in a slight different way.

<sup>2</sup>In future there may be sources which deliver XML documents for a specific domain in a standardized format. Thus in this cases complex wrappings will not be necessary anymore, only XSL-transformations have to be applied.

- *structural or intensional relevance* estimates if a source models the relevant world aspects, i.e. if the searched properties of some objects are part of the exported source data model (e.g. "Does the vendor sell car tires?"),
- *extensional relevance* estimates if a source contains the searched objects (e.g. "Does the vendor X sell car tires from manufacturer Y?"),
- *qualitative relevance* estimates who "good" a source is, i.e. gives answers to the questions whether a source contains current data or about its typical response time (e.g. "Are the offers older than one week? How often was the server not reachable?").

In case of query processing, the optimal solution is known and is described through the user's query and the current execution context, i.e. to find good relevant sources we have to minimize the distance between the query, the user preferences, the environmental information, and a group of sources. To determine the distance we utilize source descriptions, further described in Section 3.2, together with a multi-step source selection process (c.f. Sect. 3.3).

In mediator systems, the user poses queries against a global schema. There are two complementary ways to do the necessary mapping from the global schema to the local schemes a source provides [20].

In *global-as-view* schema integration, the global schema is expressed in terms of the local schemes. This bottom-up approach collects the different source schemes and integrates them through special integration techniques (syntactic and semantic conflict resolutions) into the global schema [25]. The problem with this approach is its maintenance: Adding and removing sources always leads to global schema modifications, which makes this approach nearly inapplicable in the context of the type of VirtDBS we discuss here. Systems that use this approach are e.g. TSIMMIS [9], Garlic [17], or COIN [11].

The opposite approach is the *local-as-view* schema integration, where the local schemes are expressed through terms of the global schema [33]. In this top-down approach the global schema is static and normally requires no modification when adding or removing sources from the system. Query processing techniques have to utilize complex view processing approaches [18] to find *structural and extensional relevant* sources. Systems that use this approach are e.g. Information Manifold [26] or the system described in [28].

In our local-as-view approach, relevant sources are integrated by extended distributed query processing techniques [24]. Traditional query processing is divided into three phases: query translation, query optimization, and query execution [12]. In dynamic environments like the Internet these strict phases can no longer hold up. For example if a source access fails due to server downtime another source has to be selected while processing the query, at best without affecting other parts of the query.

This technique is called *dynamic query processing* [4], i.e. intervene the current query execution process, modify the query, and continue processing. Other approaches which use some of these techniques as well are, for instance, query scrambling [34, 7], the Tukwilla system [21], or the system described in [4]. Indeed, none of these systems can cope with the permanent failure of a source; this critical situation always leads to the termination of the whole query, whereas in our approach we always try to recover a broken query.

### 3 THE DYNAQUEST-FRAMEWORK

DYNAQUEST consists of three parts. The first part defines a

way to describe the sources which later should be used. These descriptions are essential for query processing and the adaptation of the query processing: To formulate the source descriptions we developed the *source description framework (SDF)* (see Sect. 3.2).

The second part of DYNAQUEST, the *query processing and monitoring framework*, defines plan operators, which are the basic construction elements for query execution plans, and the necessary execution environment (POBOX – plan operator box). This POBOX is responsible for the controlled execution of the query and serves as a central intervention point. For space reasons we omitted the description of this part of DYNAQUEST (for further information we refer to [14]).

Finally, the last and third part of the framework embeds the previously mentioned parts into a global agent-based architecture (cf. Sect. 3.1). In the following, we will start our presentation with the architecture of DYNAQUEST.

#### 3.1 DYNAQUEST'S Agent-Based Architecture

The DYNAQUEST framework utilizes agent concepts ([37]) to build virtual database systems that support users to cope with many of the already mentioned Internet specific peculiarities. The design of our architecture was driven by the two main objectives: reduction of complexity and utilization of locality. With these central goals in mind, we hope to increase scalability of our query processing system, which should be able to cope with thousands of different sources.

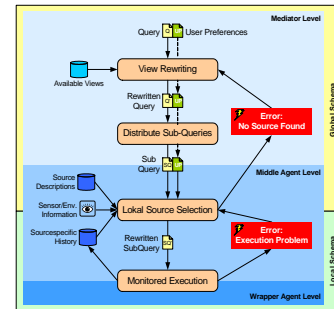


Figure 2: Global query processing in DYNAQUEST

In DYNAQUEST query processing is implemented as a two phase process (c.f. Fig. 2). In the first phase (the global source selection), standard view processing algorithms (e.g. [30]) are used to identify structural relevant *meta sources*. Meta sources are groups of similar sources which provide a common structure or common domain (e.g. car vendors). Each of these meta sources exports one or more view descriptions in terms of the global schema. No further information (for example limited source capabilities or quality information) are given in the meta data at this global level. We introduce this additional structuring level to cope with the complexity of view processing algorithms [1].

Special components, called *middle agents* [23] are responsible for managing these meta sources. They select the local sources, which should be used in the current query answering process, i.e. create and execute query sub plans applying only managed sources. In this second processing phase, not only structural, but also extensional and quality information, accounting user preferences, is treated. A further description of this local query processing can be found in Section 3.3.

Another important task relies on the middle agents: They are responsible for *recovery mechanisms*. These mechanisms

are necessary to cope with unforeseen source problems, like the slow down of a queried server or other mismatches between source descriptions and real server behavior. In our work, we concentrate on recovery functions that try to replace a corrupt source by an alternate source. The main problem with changing a data source while a query is already running is the handling of processed data. In the worst case the query has to be restarted and all processed objects have to be invalidated, which is, best to our current knowledge, the standard behavior of other approaches. This is often not acceptable. DYNAQUEST tries to reduce cases where data objects in the query tree have to be discarded and provides different recovery strategies:

- *Alternate Source*: Usage of mirror-pages with identical URL prefix.
- *Last-Delivered-Element*: The last delivered data item is stored. If a source fails and is replaced by another source no data object is passed until the last delivered element (or one that has a higher order position) has fetched. Only small modifications have to be done to the initial query processing tree, but the sources have to provide the necessary query requirements (ordered data delivery).
- *Key-Index*: This mechanism requires that there is one identifying primary key. If a source fails and is replaced by another source only those objects are passed, which are not already placed in the key index.  
This technique also allows raising the coverage of a query by querying multiple sources simultaneous and removing duplicates. Another way to use this approach is to increase potential quality by querying always both alternate sources and pass only those data that is delivered from both sources.

The installation of the recovery techniques and the decision to swap sources depends on the current query context and is handled by the middle agents.

After successfully processing the local sub queries and the transformation to the query specific global data model, the results are delivered to the *query distribution agent*, which combines the results of the different involved middle agents. The user agent receives afterwards the result and presents it in an appropriate way to the user.

### 3.2 SDF – Source Description Framework

To support the decision which source should be used for query processing, special knowledge is needed. This knowledge can be modelled by source descriptions. DYNAQUEST provides a *source description framework* (SDF which is based on RDF.

RDF provides a simple framework in which own extensions can easily be integrated. Its graph-based structure corresponds to the distributed nature of our system and it is a standard from the W3C<sup>3</sup> which in the future possibly becomes the standard for meta data exchange in the Internet. SDF/RDF fragments are easy to integrate into existing HTML pages. Supplementary, RDF has a model inherent concept of identity, by using worldwide unique URIs<sup>4</sup>. This concept makes it easy to use the same concept at different places, access remote defined information, and, by though, information that was not explicit defined for this application.

SDF defines a vocabulary to model source descriptions and provides guidance how to develop these descriptions. Essentially, there are three kinds of information modelled in the

descriptions that corresponds to the source relevance aspects already described in Section 2.

Additionally, SDF allows the description of user queries and user preferences that we will not discuss here.

In the following, we will present the steps that have to be considered to build useful source descriptions to support dynamic and adaptive query processing.

#### 3.2.1 Structural Source Descriptions

The initial step in creating source descriptions considers structural aspects, and first of all the exported schema that contains the attributes and entities the source provides via an access interface (e.g. via an html-form). The SDF uses a simple relational formalism to describe the export schema. The following fragment shows a part of a schema description in SDF:

```
<ER:Attribute rdf:ID="Manufacturer_ID">
  <rdf:type rdf:resource="&xsd:number"/>
</ER:Attribute>
<ER:Attribute rdf:ID="Manufacturer_Name">
  <rdf:type rdf:resource="&xsd:string"/>
</ER:Attribute>
<ER:Entity rdf:ID="Manufacturer">
  <ER:hasPrimaryAttribute
    rdf:resource="#Manufacturer_ID"/>
  <ER:hasAttribute
    rdf:resource="#Manufacturer_Name"/>
  ...
</ER:Entity>
<ER:Entity rdf:ID="Car">
  ...
  <ER:hasAttribute
    rdf:resource="#Manufacturer_ID"/>
</ER:Entity>
```

This SDF-fragment defines an entity *Manufacturer* that has the primary key *Manufacturer\_ID*, another attribute *Manufacturer\_Name*, and an entity *Car* with the foreign key attribute *Manufacturer\_ID*. Each attribute can be referenced from any entity, so referential integrity constraints will survive the transformation to the SDF. The referenced attribute has not even to be defined in the same description file anyway. Unfortunately, RDF provides no real scope concept. Namespaces for every entity were discarded because of their overhead (each entity has to be defined in a separate file), so we decided to prefix each attribute with the name of the owning entity. The attributes are defined with type information that uses the XML Schema<sup>5</sup> data types. Additional, information like a description of the attribute in natural speech or the relation to global concepts can easily be appended, as follows.

By now, we modelled the exported schema in the vocabulary of the source. The next step to do is to provide mappings from the source description to a global schema. The global schema forms the base to pose the user query. It need not to be one interconnected schema but can consist of many different schemes. By this, we allow to use different ontologies [15] in the query formulation process. The query poser is in charge to combine only reasonable ontologies in the query.

In our environment with changing source sets, we apply the local-as-view approach and, therefore, the direction of the schema mapping is always from the local schema to the global schema (cf. Sect. 2). In DYNAQUEST it is sufficient to map the attributes only, mappings have to be given only for those attribute which participate in the access methods of the source (see below).

The provided SDF constructs to describe the mappings can be divided into the following five classes: static one-to-one, dynamic one-to-one, dynamic one-to-many, dynamic many-to-one, and dynamic many-to-many mappings.

<sup>3</sup><http://www.w3.org/>

<sup>4</sup>Universal Resource Identifiers

<sup>5</sup><http://www.w3.org/XML/Schema>

Static mappings are used when there is no additional processing necessary to evaluate the relationship between two attributes. Dynamic mappings require the execution of a function to relate two attributes. Examples for static one-to-one mappings are equivalent-mappings<sup>6</sup>, which express that two attributes although with different names, have the same meaning, sub- or superset relationships, or overlapping, which means that the two attributes may have some values in common. The following example demonstrates that a local attribute *Author* is the same as the global attribute *creator*, which is a term from the Dublin Core element set<sup>7</sup>:

```
<ER:Attribute rdf:ID="Author">
  <rdf:type rdf:resource="&xsd:string"/>
  <daml:equivalentTo rdf:resource="&dc;creator"/>
</ER:Attribute>
```

Dynamic one-to-one mappings are used to relate attributes, which intrinsically mean the same but have to be converted to be comparable. Reasonable transformation functions are for example reformatting or unit transformations.

In SDF these dynamic functions are represented by a special node, which has a number of (ordered) input connections, a transformation function, and one output connection. The following gives an example where the global price is represented in dollar and local price in euro:

```
<SDF:transformation rdf:ID="EuroToDollar">
  <SDF:transformationInput
    rdf:resource="#carPrice"/>
  <SDF:transformationOutput
    rdf:resource="http://global/carPrice"/>
  <SDF:transformationFunction
    rdf:resource="http://globalfunc/EuroToDollar"/>
</SDF:transformation>
```

It is not always possible to create a direct one-to-one mapping between attributes of the local and the global schema. For these cases we provide one-to-many and many-to-one mappings. A one-to-many mapping describes a partitioning of an attribute, i.e. an attribute in the local schema is split up to many attributes in the global schema.

In contrast, the many-to-one mapping describes a merging of attributes; i.e. a set of attributes of the local schema is mapped to one attribute of the global schemes. The following example shows how two different date representations are related:

```
<SDF:transformation rdf:ID="DateBuild">
  <rdf:Seq rdf:ID="DateBuildInput">
    <rdf:li rdf:resource="#production_day"/>
    <rdf:li rdf:resource="#production_month"/>
    <rdf:li rdf:resource="#production_year"/>
  </rdf:Seq>
  <SDF:transformationOutput
    rdf:resource="http://global/production_date"/>
  <SDF:transformationFunction
    rdf:resource="http://globalfunc/BuildDate"/>
</SDF:transformation>
```

Dynamic many-to-many mappings are simulated by two one-to-many mappings, where the "one"-sides are combined by a special node `SDF:many_to_many`. This has to be done because of the RDF triple structure.

Because there is not always one global schema, and sources sometimes participate in different virtual databases, our SDF allows the mapping to different global schemes. This can be done by adding additional mapping descriptions. In the following example, where we added a Marc21 [8] description tag for authors.

```
<ER:Attribute rdf:ID="Author">
  <rdf:type rdf:resource="&xsd:string"/>
```

<sup>6</sup>for which we use `daml:equivalent` (<http://www.daml.org>)

<sup>7</sup><http://www.dublincore.org/>

```
<daml:equivalentTo
  rdf:resource="&dc;creator"/>
<daml:equivalentTo
  rdf:resource="&marc21;600_10_\$a"/>
</ER:Attribute>
```

Web sources normally allow only very restricted access to their content — queries often can only be posed through a web form. This leads to peculiarities that e.g. only restricted combinations of attributes are allowed as input values or some of the attributes need a value binding.

These restrictions are expressed by access patterns [6], specifying which input attribute combination yields to which output attribute combination. To prevent initial overhead for the creation of these patterns and to reduce maintenance efforts, so called *adornments* [10] are used to express that specific attributes must have an value binding, are optional, or have to use values from a fixed set of entries. In our approach, we use similar adornments like [10].

### 3.2.2 Extensional Source Descriptions

In opposite to this straightforward definition of structural aspects, source extensions are very hard to describe. Not only the huge data volume we have to cope with is a problem, but also there is not always a way in get access to all stored data in the source. For this reasons SDF provides only simple mechanisms to describe the extension of sources by predicates, using operands which are defined in a global ontology (e.g. `Manufacturer_ID>1000` and `Manufacturer_ID<10000`).

Because of this inexactness in extensional source selection, the dynamic and adaptive query processing techniques provided by DYNAQUEST are a fundamental requirement and unburden the user from cases where sources do not return the expected results. In such a case, the system has to proceed with another alternate source.

### 3.2.3 Quality Source Descriptions

The final part, the quality descriptions, is used to select from a set of intensional and extensional similar sources. These quality descriptions, which are modelled as attribute value pairs, can be divided into source and query specific aspects. The source specific aspects describe the whole source (e.g. availability) and query specific aspects describe the properties of a input-output-pattern combination (e.g. the response time, or the coverage). Due to the RDF roots of SDF, it is easy to bind these quality descriptions to specific subparts of the source description (query description nodes). SDF constructs used to model quality descriptions cover technical criteria (e.g. availability, date deliver rate, security, monetary costs, last update time, update frequency etc.), extensional criteria (e.g. correctness, consistency, coverage/completeness, timeliness etc.), and supplier specific criteria (e.g. reputation, objectivity etc.).

The quality description can be further divided into global, aggregated descriptions, which are stored at the wrapper or middle agent, and user specific descriptions, which are stored in the user profile. This has to be done because the value of some criteria, like reputation, depends on the users rating.

### 3.2.4 Summary

To summarize, SDF provides the possibility to describe distributed sources with limited access pattern. These descriptions are used to determine the source relevance concerning a specific query with regard to intensional, extensional, and quality aspects, and support the source selection process. Other works on source descriptions, like [35] or [27]

are less detailed and address in particular no quality aspects. In contrast, [29] regards quality aspects, but ignore user specific aspects. Additionally, to our best knowledge, so far proposed source descriptions depend on one single global (none distributed) schema and allow no multi mapping to different global schemes.

### 3.3 Local query processing

For local query processing, there has to occur a source selection. This process has to consider all relevance aspects described in Section 2, i.e. the result of a selection process should contain only sources which are structural, extensional, and qualitative not less relevant than specified by a user's individual preferences. In DYNAQUEST the middle agents are responsible for source selection.

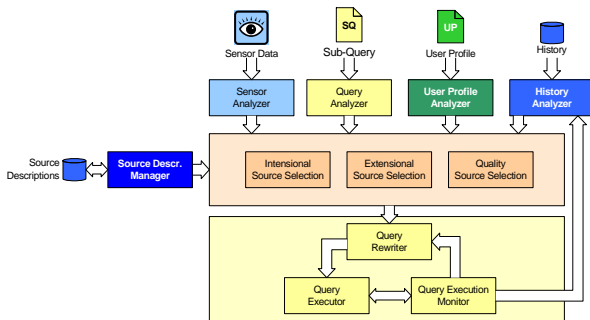


Figure 3: Architecture of a Middle Agent

Figure 3 shows the architecture of a DYNAQUEST middle agent and the different data sources that are utilized to do the selection process. In detail, the following information is needed to decide the relevance of a specific source:

- The *query* has two parts, which are important. The attributes are used to check structural relevance and the predicates give further hints about the extensional relevance of a source.
- The *user profile* is employed to enhance the quality of the set of considered sources. The user profile considers three aspects that are relevant for query processing in virtual databases relevant.
  - *attribute weighting*: In this part of the profile a user can specify, if some of the queried attributes are of higher importance than other or even just "nice-to-have-attributes". By this, we are able to rank the sources respecting the user attribute weights.
  - *quality expectations*: This part of the user profiles contains quality expectations, e.g. if the user is willing to wait for a while for good results, or if he is not willing to pay for it. These weightings are soft constrains; i.e. a source selection process does not break, if these expectation cannot be fulfilled, but may decrease the quality of the result.
  - *User specific, subjective source valuations*: Values in this part of the user profile always overwrite specific quality source descriptions. This is necessary to comply with specific user opinions concerning special sources.
- *Sensor-data* and *history data* is used to improve the final quality ranking of the sources and to test specific user criteria, like fast processing of a query. If a source typically

does have bad data delivery rate every Tuesdays from eight to ten and the current time is Tuesday nine o'clock, it may be more effective to use an alternate source. Technically, this historic data can be analyzed by neural networks, specific data mining techniques (time series analysis) or the WebPT-Approach [7].

- To rank the sources concerning the above described criteria the *source descriptions* (c.f. Sect. 3.2) are used. A special component, the *source description manager* is responsible for the management, access and maintenance of the source descriptions. The SDF-descriptions are compiled into a source description database to allow efficient access.

#### 3.3.1 Local Source Selection

The local sources selection traverses three stages. In the first stage, the *structural relevance check*, local sources are selected, which provide at least one of the requested attributes. This can be achieved very efficient by using an attribute source index. Afterwards the sources are ranked respecting the user preferences concerning the attribute weighting. Non-applicable source are discarded. The remaining sources  $S_i$  are ranked with the aid the following simple formula:

$$v(S_i) = \sum_{j=1}^n sc(S_i, a_j) * uw(a_j), \text{ with}$$

$$sc(S_i, a_j) = \begin{cases} 1 & : a_j \in StructDescr(S_i) \\ 0 & : else \end{cases}$$

$n$  the number of attributes in the query, and  $uw(a_j)$  returning the user specific weighting of the query-attribute  $a_j$ .

This ranking is the input to the next stage, the *extensional relevance check*. In this stage we try to reorder the ranking based on the probability that a source contains relevant extensions or discard sources which are surly extensional irrelevant.

Finally, the last part of the selections process concerns information quality. Currently, we examine techniques from the field of multi-criteria analysis [38] to support this phase of the source selection: The user's desired (query specific) information quality must be considered when selecting and accessing sources.

As we already sketched in Section 1, information quality can be seen as an aggregate deduced from a set of different criteria. Possible criteria are, for instance, time constraints, the completeness of information, or the maximum costs for queried information, like modelled in the quality part of SDF. At each case, information quality must be interpreted with respect to such criteria, the users preferences regarding this criteria and source descriptions. The ranking acquired so far is integrated in the processing as one special criterion (attribute occurrence).

To cope with these quality-oriented aspects we utilize the multi-criteria decision software ProDecX that we developed in the past. The core of this software tool is based on a variant of the PROMETHEE (= Preference Ranking Organization METHod for Enrichment Evaluations) method [5] and can be categorized as an outranking method [38]. Compared to the basic PROMETHEE algorithm, the implementation of ProDecX is extended to allow the explicit consideration of data and weight uncertainty. Thus, the tool can be used to process very intuitive descriptions for alternatives (sources) which are allowed to be fuzzy (e.g. the response time of sources can be described by probability distributions). In addition, the weighting of criteria can be specified in this way (e.g.

to describe the (relative) relevance of time constraints). The parameterization is the foundation for a simple stochastic simulation and the results are compiled as a ranking of possible alternatives.

As the result of the three query selection phases, we obtain a ranking of the sources, which promise the best applicability in the current query context. The highest ranked source is selected, possibly supplemented by recovery mechanisms 3.1, integrated in the plan and finally monitored executed.

#### 4 CONCLUSION AND FUTURE WORK

In this work we presented our framework DYNAQUEST. This framework simplifies the development of robust, high-available, fault-tolerant and scalable virtual databases, respecting user specific (quality) concerns. At this, DYNAQUEST focuses on query processing aspects in highly distributed and dynamic environments. Source selection and source switching is supported by source descriptions based on SDF and source wrapping middle agents. First source selection methods were presented by using the ProDecX software, which is currently extended to the special needs of DYNAQUEST.

Currently, we are implementing main parts of the framework to evaluate the different techniques we introduced. Furthermore, we have to extend the source description framework to cope with more complex sources and integrate the source description framework into the source selection process, evaluate the utilization of ProDecX, and further investigate the application of case-based-reasoning methods for the detection of critical situations in the context of DYNAQUEST. Afterwards, we are going to test our framework within an electronic commerce application concerning car vendors.

#### References

- [1] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 254–263, Seattle, Washington, 1998. ACM Press.
- [2] Fabien Azavant and Arnaud Sahuguet. W4f: a wysiwyg wrapper factory for minute-made wrappers. Technical report, PENN Database Research Group, 1998.
- [3] Greg Barish, Craig A. Knoblock, Yi-Shin Chen, Steve Minton, Andrew Philpot, and Cyrus Shahabi. Theaterloc: A case study in building an information integration application. In *IJCAI Workshop on Intelligent Information Integration*, Stockholm, Sweden, 1999.
- [4] Luc Bouganim, Francosie Fabret, C. Mohan, and Patrik Valduriez. A dynamic query processing architecture for data integration systems. *Bulletin of the Technical Committee on Data Engineering*, 23(2):42–48, 2000.
- [5] J.P. Brans and B. Mareshal. The promethee methods for mcdm; the prom-calc, gaia and bankadviser software. In C.A. Bana e Costa, editor, *Readings in Multiple Criteria Decision Aid*, pages 216–252, 1990.
- [6] Daniela Florescu, Alon Levy, Ioana Manolescu, and Dan Suciu. Query optimization in the presence of limited access patterns. In *SIGMOD 1999*, pages 311–322, Philadelphia, 1999. ACM.
- [7] Michael J. Franklin, George A. Mihaila, Louiqa Raschid, Tolga Urhan, Maria Esther Vidal, and Vladimir Zadorozhny. Searching and querying wide-area distributed collections. In *Russian National Conference on Digital Libraries*, St. Petersburg, 1999.
- [8] B. Furrie. *Understanding Marc Bibliographic: Machine-Readable Cataloging*. Library of Congress, 6th edition (january 2001) edition, 2001.
- [9] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey Ullman, Vasilis Vassalos, and Jennifer Widom. The tsimmiis approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.
- [10] Hector Garcia-Molina and Ramana Yerneni. Coping with limited capabilities of sources. In Alejandro P. Buchmann, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), GI-Fachtagung*, Informatik Aktuell, pages 1–19, Freiburg, 1999. Springer.
- [11] C.H. Goh, S. Bressan, S.E. Madnick, and Michael Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–293, 1999.
- [12] Gotz Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–170, 1993.
- [13] Marco Grawunder. Agentenbasierte adaptive und dynamische anfrageverarbeitung in virtuellen datenbanksystemen. Forschungsbericht 2001-14, Technische Universität Berlin, 11./12.Oktober 2001 2001.
- [14] Marco Grawunder and Frank Köster. DYNAQUEST. Internal Report, University of Oldenburg, Escherweg 2, 2002.
- [15] Michael Gruniger and Jintae Lee. Ontology applications and design - introduction. *Communications of the ACM*, 45(2):39–41, 2002.
- [16] Ashish Gupta, Venky Harinarayan, and Anand Rajarman. Virtual database technology. In *Proceedings of the Fourteenth International Conference on Data Engineering*, pages 297–301, Orlando, Florida, USA, 1998.
- [17] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 276–285, Athens, Greece, 1997.
- [18] Alon Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
- [19] Joachim Hammer, Hector Garcia-Molina, Svetlozar Nestorov, Ramana Yerneni, Marcus Breunug, and Vasilis Vassalos. Template-based wrappers in the tsimmiis system. In *SIGMOD*, pages 532–535, Tucson, Arizona, USA, 1997.
- [20] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Principles of Database Systems (PODS)*, pages 51–61, Tuscon, Arizona, USA, 1997. ACM.
- [21] Zachary G. Ives, Daniela Florescu, Marc Friedman, Alon Levy, and Daniel S. Weld. An adaptive query execution system for data integration. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 299–310, Philadelphia, Pennsylvania, USA, 1999. ACM Press.
- [22] Matthias Jarke, Maurizio Lenzerini, Y. Vassiliou, and P. Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2 edition, 2002.
- [23] Matthias Klusch and Katia Sycara. Brokering and matchmaking for coordination of agent societies: A survey. In Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors, *Coordination of Internet Agents - Models, Technologies, and Applications*, pages 197–224. Springer-Verlag, 2001.
- [24] Donald Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469, 2000.
- [25] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Principles of Database Systems PODS 2002*, page 14, Madison, Wisconsin, USA, 2002. ACM.
- [26] Alon Levy. The information manifold approach to data integration. *IEEE Intelligent Systems*, 13(5), 1998.
- [27] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd VLDB Conference*, pages 251–262, Bombay, Indien, 1996.
- [28] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. Answering xml queries over heterogeneous data sources. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases*, Roma, Italy, 2001. Morgan Kaufmann.
- [29] George Mihaila, Louiqa Raschid, and Maria Esther Vidal. Using quality of data metadata for source selection and ranking. In Dan Suciu and Gottfried Vossen, editors, *Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000*, pages 93–98, Adam's Mark Hotel, Dallas, Texas, USA, 2000.
- [30] Rachel Pottinger and Alon Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB Journal*, 10(2-3):182–198, 2001.
- [31] Amit Sheth and James Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
- [32] Devika Subramanian, Russel Greiner, and Judea Pearl. The relevance of relevance. *Artificial Intelligence*, 97(1-2):1–5, 1997.
- [33] Jeffrey Ullman. Information integration using logical views. In *Database Theory - ICDT '97, 6th International Conference*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40, Delphi, Greece, 1997. Springer.
- [34] Tolga Urhan, Michael J. Franklin, and Laurent Amsaleg. Cost-based query scrambling for initial delays. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD*, pages 130–141, Seattle, Washington, 1998. ACM.
- [35] Vasilis Vassalos and Yannis Papakonstantinou. Describing and using query capabilities of heterogeneous sources. In *VLDB*, pages 256–265, Athen, Griechenland, 1997. ACM Press.
- [36] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.
- [37] Michael Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2001.
- [38] H.-J. Zimmermann and L. Gutsche. *Multi-Criteria Analyse*. Springer, 1991.