Planning and Optimizing Semantic Information Requests Using Domain Modeling and Resource Characteristics

Shuchi Patel and Amit Sheth

Large Scale Distributed Information Systems (LSDIS) Lab
Department of Computer Science, University of Georgia, Athens, GA 30602
{shuchi,amit}@cs.uga.edu
http://lsdis.cs.uga.edu

Abstract. The focus of information integration systems providing single access to data from distributed, diverse, and autonomous sources (including web-based sources) has changed from syntax and structure to semantics, allowing more meaningful integration of data. InfoQuilt[1] goes one step further to support knowledge discovery by providing users with tools to analyze the data, understand the domains and relationships between them, and explore new potential relationships. It provides a framework to model the semantics of domains, complex semantic relationships between them, characteristics of available sources, and provides an interface to specify information requests that the system can "understand". This thesis focuses on the use of knowledge about domains, their relationships, and sources to efficiently create practical execution plans for such semantic information requests.

1 Introduction

The amount of "data" available has exploded immensely in the last few years. The sources of this data, including the traditional file systems, databases, web-based sources, etc., are autonomous repositories that were developed independently. They therefore have different models for same domains. Most tools available to search information from web-based sources provide pure keyword based search that is not powerful enough to describe a user's information need. For instance, how would you describe the following request to a search engine such that it knows precisely what you are asking for? *"Find information about all earthquakes that occurred after 1990 in India resulting in death of more than 1000 people."* Also, the results generated have a low precision and recall. The condition of finding too much, and often irrelevant, data in response to any information need, whether using a search or browsing, is known as the problem of *information overload*.

[1] One of the incarnations of the InfoQuilt system, as applied to the geographic information as part of the NSF Digital Library II initiative is the ADEPT-UGA system [1]

C. Batini et al. (Eds.): CoopIS 2001, LNCS 2172, pp. 135–149, 2001.

Providing access to diverse multiple sources via a common interface has been an interesting research problem called *information integration*. Sheth [19] describes different types of heterogeneities to be dealt with and related issues for achieving interoperability. Use of metadata, especially domain specific metadata, to address this issue is interesting and has gone a long way. [20] presents several examples. However, the need to semantically relate data to make sense out of it is what makes the problem more interesting and challenging. Consider the following information request. *"Find all nuclear tests conducted by USSR or US after 1970 and find all earthquakes that could have occurred as a result of the tests."* Among other things, the system needs to understand how a nuclear test can cause an earthquake. This requires understanding of semantics of the domains and the interaction between them. Structured databases that mainly focus on syntax and structure of data do not support this.

Very frequently, users use functions to post-process and analyze data (e.g. statistical analysis techniques). Of special interest are simulation programs that are used for forecasting, extrapolation, etc. Consider the following request. *"Forecast the urban growth pattern in the San Francisco bay area for the next 10 years in time steps of 1 year based on roads, slopes and vegetation patterns in the area."*

Some of the important issues in an information integration system are:

- Modeling the domains and relationships between them
- Resolving heterogeneities between the sources [19]
- A powerful interface to specify information requests (beyond traditional keyword queries and queries against structured database as in SQL) that can semantically describe the information need
- Scalability of the system
- Efficient planning and optimization

InfoQuilt addresses several of these issues. Several other research efforts such as [4], [13], [15], and [10] have developed approaches to represent the semantics of domains and the characteristics of sources. Additionally, InfoQuilt provides frameworks to model complex inter-domain relationships and information requests that capture the semantics of an information need. The vision of the InfoQuilt system is different from most other information integration systems. Our goal is to allow users to analyze the data available from a multitude of diverse autonomous sources, gain better understanding of the domains and their interactions and determine information leading to decision making via analysis of potential semantic relationships between the data provided by different sources. This support for knowledge discovery is a novel feature of InfoQuilt.

This paper focuses on planning and optimization of information requests, which are known as Information Scapes or IScapes in InfoQuilt. The algorithms described have been implemented and the examples presented have been tested. Given an IScape, the system creates an execution plan consisting of queries against relevant sources and steps to integrate the data. Creation of high-quality near-optimal plans is crucial due to the following main reason. Web sources constitute a large portion of available sources. Retrieving data from them over the network using wrappers is slow relative to a source that is a database. The query

execution is therefore relatively slow. However, certain semantic optimizations allow us to choose one execution plan over another so that the execution is faster. The following are key features of our algorithm:

- Efficient source selection - excluding sources that will not provide useful results or will provide redundant data
- Generation of executable plans which respect the limitations of resources
- Use of sources in conjunction to be able to use resources with query capability limitations and missing data
- Integration of data retrieved from the sources selected, including relationship evaluation and post-processing (e.g. evaluating functions and running simulations)

Section 2 briefly describes how the knowledge base of the system is represented, how IScapes are represented, and how InfoQuilt supports an environment for knowledge discovery. Section 3 describes planning of IScapes and optimization of the execution plans. Section 4 discusses the runtime architecture of the system and IScape execution. We compare InfoQuilt with other systems in sect 5. Section 6 presents our conclusions and directions for future work.

2 Background

In a typical scenario in using the InfoQuilt system, an administrator first identifies the domains of interest and models them. Section 2.1 describes domain modeling. Next, he models complex relationships between the domains as described in Sect. 2.2. He then identifies data sources for those domains, creates their wrappers, and models them as described in Sect. 2.3. A distinguishing feature of InfoQuilt is its ability to use functions. We describe them in sect 2.4. The knowledge about ontologies, relationships, resources and functions forms the *knowledge base* of the system. The administrator uses a graphical tool called the *Knowledge Builder* (KB) to easily create and maintain the knowledge base. Users can then create IScapes as discussed in Sect. 2.5 using a visual tool called the *IScape Builder* (IB). Section 2.6 shows how InfoQuilt can help users discover knowledge. Details left out in this paper due to space restriction can be found in [17]. Additional detailed discussion on the knowledge base, KB, IScapes, IB, and the support for knowledge discovery can be found in [21].

2.1 Domain Modeling

One form of heterogeneity between different sources that needs to be resolved is the difference in their views of the domain. InfoQuilt uses ontologies for this. An *ontology* captures useful semantics of the domain such as the terms and concepts of interest, their meanings, relationships between them, and characteristics of the domain. The domain characteristics are described as a set of domain rules and functional dependencies. We will use the terms "ontology" and "domain" interchangeably in the rest of the paper unless explicitly specified.

Example 1. Consider the domain of nuclear tests. Some related terms are test site, explosive yield, seismic body wave magnitude, type (underground, atmospheric, etc.), latitude, longitude, etc. The magnitude of a test is in the range 0 to 10.[2] We represent this as domain rules. Also, given a test site, the latitude and longitude are the same. We represent this as a functional dependency (FD).

```
NuclearTest( testSite, explosiveYield, waveMagnitude, testType,
             eventDate, conductedBy, latitude, longitude,
             waveMagnitude > 0, waveMagnitude < 10,
             testSite -> latitude longitude );
```

We will use the above notation to represent ontologies in the paper. □

2.2 Inter-ontological Relationships

Real world entities are related to each other in various ways. These relationships can be simple , for e.g., a car "*is a*" vehicle, or complex, for e.g., an earthquake "*causes*" a tsunami. InfoQuilt is novel in its ability to model such relationships.

Example 2. Consider the relationship between a nuclear test and an earthquake. We use the `NuclearTest` ontology from e.g. 1 and model earthquake as follows:

```
Earthquake( eventDate, description, region, magnitude, latitude,
            longitude, numberOfDeaths, damagePhoto,
            magnitude > 0 );
```

We can say that a nuclear test could have "*caused*" an earthquake if the earthquake occurred "*some time after*" the test was conducted and "*in a nearby region*". Here, "*some time after*" and "*in nearby region*" are user-defined functions used as specialized operators as described in Sect. 2.4. For now, assume that there are two functions called `dateDifference` and `distance` that compute the difference between two dates and the distance between two locations (specified by their latitudes and longitudes) in miles respectively. We can now represent the relationship as follows:

```
NuclearTest Causes Earthquake <=
dateDifference(NuclearTest.eventDate, Earthquake.eventDate) < 30
AND distance(NuclearTest.latitude, NuclearTest.longitude,
               Earthquake,latitude, Earthquake.longitude) < 10000
```

The values 30 and 10000 are arbitrary. □

[2] The body wave magnitude does not have upper or lower bounds. However, all nuclear tests as well as earthquakes measured to date had a magnitude less than 10 and those with magnitudes less than 0 are very small. We use these bounds solely to demonstrate how the domain rules can be modeled and used.

2.3 Source Modeling

The system needs to resolve heterogeneities between the sources. This is done in part by creating an ontology for each domain and describing the sources in terms of them. This approach is known as source-centric approach. Since the sources are modeled completely independent of each other, they can be easily added, removed and re-modeled without changing the ontologies. The source models describe their characteristics and query limitations in terms of the following:

Resource Rules: Resource or Data Characteristic (DC) rules describe conditions that always hold true for data retrieved from resource.

Example 3. Consider a resource that lists earthquakes in California after January 1, 1980. We can specify this by the rules:

```
region = "California"
eventDate >= "January 1, 1980"
```

□

Local Completeness: Local Completeness (LC) rules describe a subset of the domain for which the resource is known to have complete information.

Example 4. Ontology `DirectFlight` models direct flights from one US city to another. The database resource at Atlanta Airport provides data on *every* flight departing from and arriving at Atlanta. We specify this using the LC rules:

```
toCity = "Atlanta"
fromCity = "Atlanta"
```

□

Binding Patterns: Web sites that allow users to search their database(s) via HTML forms using CGI scripts, servlets, etc. often require the user to provide values for certain parameters. The query answering system needs to provide values for these parameters to use it. It is crucial for the system to be aware of these limitations. These are represented as binding patterns (BP).

Example 5. The AirTran Airways (*http://www.airtran.com*) web site requires the user to specify source, destination, dates of travel, etc. We represent this as the following BP:

```
[toCity, toState, fromCity, fromState,
departureMonth, departureDay]
```

□

2.4 Functions and Simulations

Example 2 demonstrates use of two functions to create special operators used to define a relationship between `NuclearTest` and `Earthquake`, which cannot be expressed by using only relational and logical operators. Another use of functions

is to post-process data. For e.g., use of various statistical analysis techniques to analyze data is very frequent. Of particular interest, are simulations. For e.g., researchers in the field of Geographical Information Systems (GIS) use them to extrapolate patterns of urban growth, deforestation, etc. based on models. These are available as independent programs. They can be used over the data retrieved from available sources (assuming that the system has access to appropriate sources) and the result can be presented to the user.

InfoQuilt views such functions and simulations as important sources of related information. It maintains a declarative description of the functions in a *Function Store*, a component of its knowledge base. The administrator is responsible for providing an implementation for them.

2.5 Information Scapes (IScapes)

InfoQuilt is unique in its ability to answer complex information requests known as *Information Scapes* or *IScapes*. Since they are specified in terms of the components of the knowledge base of the system, they are better able to model the semantics of a user request and the system is able to "understand" them.

Example 6. Consider the following IScape (described here as text).

"Find all nuclear tests conducted by USSR or US after 1970 and find information about earthquakes that could have occurred due to them."

Of specific interest is that the system needs to understand what the user means by "*earthquakes that could have occurred due to these tests*". This is where knowledge about the inter-ontological relationship "nuclear test causes earthquakes" available in the knowledge base of the system is useful. □

IScape also abstracts the user from the characteristics, structure, access mechanisms, etc. of the actual data sources available to the system (eventually used to answer the IScape), and their heterogeneities.

We provide a graphical toolkit, known as *IScape Builder*, that allows users to easily create IScapes step-by-step, execute them and further analyze the data. See [21] for details.

2.6 Support for Knowledge Discovery

InfoQuilt helps users discover knowledge by providing tools to analyze the data available from diverse distributed sources, gain understanding of the domains and their relationships, explore possibilities of new relationships, explore trends to support such potential relationships, and provide further insight into additional aspects about existing relationships. A detailed discussion with examples appears in [21].

3 Planning and Optimization

We now describe the creation of execution plans for IScapes and optimizations possible due to the knowledge base that InfoQuilt maintains. The sources are described in terms of the ontologies. Hence, source descriptions are similar to views defined on them. The problem of creating an execution plan is thus closely related to the problem of answering queries using views [23], [12], [7]. It was shown to be NP-complete in [12]. The algorithm we describe here uses domain and source characteristics and is therefore tractable and efficient.

A more detailed explanation of plan generation can be found in [17]. We use the following example IScape and show how its execution plan is generated.

"*Find nuclear tests conducted by USSR after January 1, 1980 with magnitude greater than 5 and earthquakes that could have occurred due to these tests.*"

We use the ontologies `NuclearTest` and `Earthquake` and the relationship between them `NuclearTest Causes Earthquake`. We use a similar notation for sources as the one we use for ontologies. We prepend `[dc]` or `[lc]` to distinguish between DC and LC rules. Following sources are available to the system.

```
NuclearTestsDB(testSite, explosiveYield, waveMagnitude,
               testType, eventDate, conductedBy,
               [dc] waveMagnitude > 3,
               [dc] eventDate > "January 1, 1985");
NuclearTestSites(testSite, latitude, longitude);
SignificantEarthquakesDB(eventDate, description, region,
                         magnitude, latitude, longitude,
                         numberOfDeaths, damagePhoto,
                         [dc] eventDate > "January 1, 1970");
```

`NuclearTestsDB` is a database of nuclear tests of magnitude greater than 3 conducted after January 1, 1985. `NuclearTestSites` provides exact locations of nuclear test sites in terms of latitudes and longitudes. `SignificantEarthquakesDB` is a database of significant earthquakes occurring after January 1, 1970.

Step 1 (Check Semantic Correctness). The first step is to check if the IScape is semantically valid by comparing the IScape constraint with the domain rules defined on the ontologies involved. The constraint in our IScape is:

```
NuclearTest.waveMagnitude > 5 AND NuclearTest.conductedBy = "USSR"
AND NuclearTest.eventDate > "January 1, 1980"
AND dateDifference(NuclearTest.eventDate,
                   Earthquake.eventDate) < 30
AND distance(NuclearTest.latitude, NuclearTest.longitude,
             Earthquake,latitude, Earthquake.longitude) < 10000
```

The system maps the relationships used in the IScape into sub-constraints. Comparing the constraint with the domain rules defined on the ontologies, we see that the IScape is semantically valid.

If the IScape enquired about nuclear tests with magnitude greater than 12 instead of 5, the domain rule "`waveMagnitude < 10`" would imply that the IScape is semantically invalid. This aspect of our algorithm is supported by various systems that support semantic integrity constraints.

Step 2 (Source Selection). Next, the planner uses domain rules and FDs of ontologies and DC rules, LC rules, and BPs of the sources to select relevant sources by applying the following rules for each ontology:

Locally Complete Sources : If there exists a source that is locally complete for some subset A of the domain of the ontology such that the part of the IScape's result that comes from that domain is a subset of A, other sources need not be used (unless the selected source has some attributes missing or a BP) since the LC rule on the source implies that using any additional sources will not provide any extra information. No source in our example are locally complete.

Non Locally Complete Sources : If a locally complete source could not be found, then the planner considers all the sources. The sources whose DC rules falsify IScape's constraint are then eliminated since they will not return any useful results for the IScape (they will get filtered by the IScape constraint).

For our example, the planner selects all sources for both ontologies and none of them are eliminated. If `NuclearTestsDB` provided data for tests conducted *before* January 1, 1980, then the rule "`eventDate < January 1, 1980`" would falsify IScape's constraint and the source would be eliminated.

Binding Patterns (BP) : Planner then decides how the values for the BPs on resources, if any, will be supplied. It can be (a) from the IScape constraint, (b) from attributes in other ontologies or (c) by using another source (associate) of the ontology in conjunction to retrieve some arbitrary values for the attribute. None of the resources selected for our example have BPs. Refer to [17] for an example that uses resources with BPs.

Associate Resource to Supply Values for Missing Attributes : It is common to come across sources that do not provide all the attributes needed. If a source has one or more attributes missing, the planner can use the FDs on the ontology to couple it with an associate resource to retrieve values of those attributes for as many entities as possible. This is done by equating the values of attributes in the LHS of the FD to join the data retrieved from the main and associate resources. System can thus deduce more information by using sources in conjunction. The attributes are equated using a default equality computing function defined for it in the function store or an exact match if no function was defined. Use of functions is necessary as it is highly unlikely that all sources will have exact same values for the attributes for a given entitiy (syntactic heterogeneity) [11]. For e.g., a nuclear test site available from one source could be "Nevada Test Site, Nevada, USA" and that from another source could be "Nevada Test Site, NV, USA". The two are semantically equal but syntactically unequal. Functions can be used to compute a context-sensitive fuzzy equality.

`NuclearTestsDB` has two missing attributes - `latitude` and `longitude`. The planner uses the FD "`testSite -> latitude longitude`" and `NuclearTestSites` as an associate resource to retrieve their values. A function `testSiteEquals` is used to equate the values of `testSite` from the two resources. Most attributes are missing from `NuclearTestSites` and there is no FD that can be used to retrieve thier values. Hence, it is eliminated it as a primary source.

The plan may use a source more than once. For e.g., a source may be used as a primary source and an associate source. The planner identifies such sources and optimizes the plan by creating a single Resource Access node.

Figure 1 shows the plan for our IScape. It shows how some sub-conditions can be pushed to sources and how some sub-conditions may be eliminated using DC rules.

`NuclearTestsDB` is joined with `NuclearTestSites` to provide values for the missing attributes - `latitude` and `longitude`. There is no check for the sub-constraint `NuclearTest.eventDate > "January 1, 1980"` because only data on tests conducted after January 1, 1985 is available from `NuclearTestsDB`.

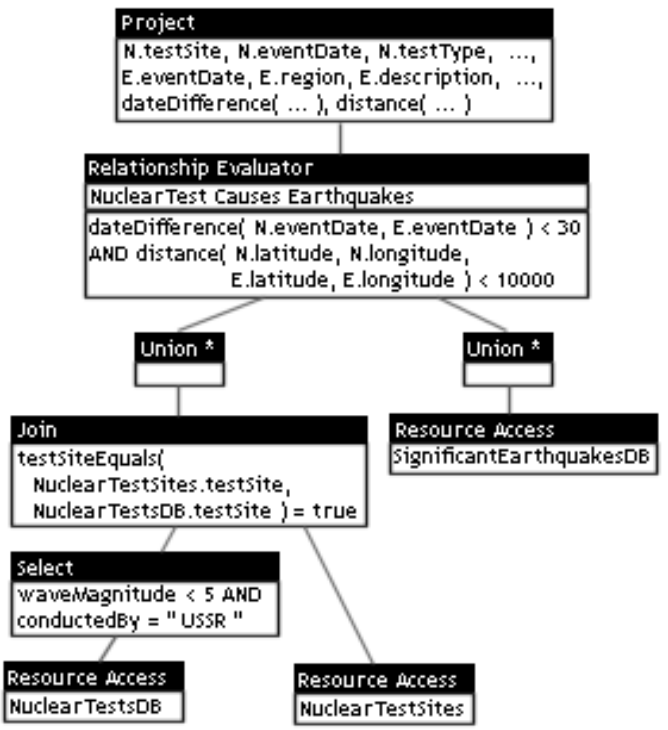

Fig. 1. IScape Execution Plan

Step 3 (Integrate Data For Each Ontology). Next, the planner creates the nodes that describe how the data retrieved from the sources is to be integrated. The first step in doing this is to create unions of results retrieved from sources selected for each ontology in the IScape.

For this, it first adds a node to compute an intermediate union of data from those sources that do not retrieve values for their BP from attributes of other ontologies. A source (of another ontology) that needs values of some attribute(s) of this ontology for its BP can then retrieve them from the intermediate union. This avoids deadlocks. We use a * in Fig. 1 to denote a union node of this type. The planner then adds another union node that computes the final union of data from all sources for the ontology. At all stages of plan creation, if a node is found to be trivial, it is eliminated.

Step 4 (Supplying BP Values from Other Ontologies). After creating the unionsNext, the planner goes through updates all the BPSupplier nodes (used with for sources that retrieve BP values from other ontologies) in the plan by creating links from it to the check that there exists an intermediate union nodes for of the ontologies from which it retrieves BP values are to be retrieved. If it does not, the source that the BPSupplier node supplies BP values to cannot be used. If the intermediate unions do exist, the planner updates the plan to point to them. See [17] for details and example.

Step 5 (Relationship and Constraint Evaluations). Next, for each ontology, all the functions and sub-constraints that involve attributes of only that ontology are evaluated. The data from different ontologies should then be integrated by evaluating the relationships used in the IScape or by creating joins if some ontology is not a part of any relationship. All remaining functions and constraints (that involve attributes of multiple ontologies) can be evaluated next.

Step 6 (Aggregation and Final Projection). The results are then grouped as specified, aggregations are computed and group constraints are evaluated. Our IScape does not require grouping of results. Finally, the result can be projected on the operands (attributes, functions and aggregates) in the projection list of the IScape. This completes the generation of the execution plan for the IScape.

4 IScape Execution

This section discusses how an IScape submitted to the InfoQuilt system is processed. We start by describing the runtime architecture.

4.1 Runtime Architecture

InfoQuilt uses an agent-based brokering architecture shown in Fig. 2. Although the agent-based architecture itself is not unique (e.g., see [5]), the capabilities of the agents to support complex relationships, multiple ontologies and IScapes differentiate the system from previous agent-based systems. This work is built

upon the work done in [18], [6], and [16]. The system provides a visual tool to monitor the execution. See [17] for details.

The Knowledge Agent acts as an interface to the knowledge base of the system represented by "Knowledge" in Fig. 2. The Broker Agent is responsible for brokering requests from other agents and co-ordination of IScape processing. All agents interact through the Broker Agent. The Resource Agents provide a uniform interface to access all resources. There is one Resource Agent per resource in the system. They address issues related to access mechanisms, mapping data from the source views to global views of the system, etc. The processing of an IScape proceeds as follows:

1. The User Agent sends the Iscape to the Broker Agent for processing.
2. The Broker Agent sends the Iscape to the Planning Agent.
3. The Planning Agent creates a plan for the Iscape as described in Sect. 3, enquiring about the ontologies, relationships, etc. from the Knowledge Agent. If the IScape is semantically invalid, it informs the Broker Agent and aborts.
4. The Planning Agent returns the plan to the Broker Agent, which responds back to the User Agent skipping steps 5-7 if plan generation was aborted.
5. The Broker Agent then sends the plan to the Correlation Agent.
6. The Correlation Agent executes the plan. It accesses the necessary resources (through Resource Agents) through the Broker Agent. We describe it in detail in sect 4.2.

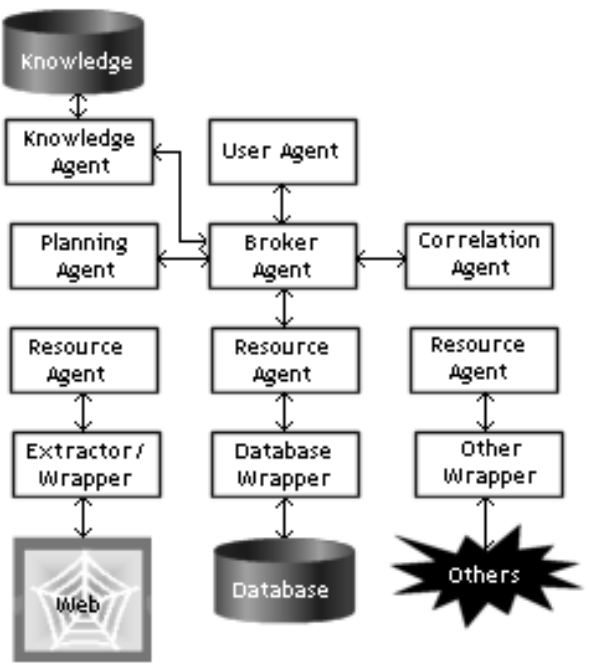

Fig. 2. InfoQuilt Runtime Architecture

7. The Correlation Agent returns the final result to the Broker Agent.
8. Finally, the Broker Agent forwards the result to the User Agent.

4.2 Multi-threaded Execution of IScape by Correlation Agent

IScape execution exploits the parallelism in the plan and is multi-threaded. The Correlation Agent starts by creating a *node processor* for each node in the plan. Each node processor is a separate thread. Every node is responsible for starting the node processors from which it expects any inputs (result sets). It then waits for all its inputs and starts processing as soon as they are available.

4.3 Execution of Functions and Simulations

Evaluation of functions (including execution of simulations) is done using Java's reflection package to pass arguments, invoke the function and retrieve the results back. The administrator is responsible for supplying the methods as static functions in Java classes.

However, simulations can be tricky. They could be executable files, scripts that need special interpreters, etc. Also, they may expect their inputs in different ways. Considering the diversity that can be encountered, the framework provided may not be sufficient for all kinds of simulation programs available. This needs further investigation. However, a large part of these can be supported using the current framework. For e.g., we use Clarke's Urban Growth Model [8] to predict urban growth in an area. The program is available as an executable and needs an input configuration file from which it reads in its parameters.

5 Related Work

SIMS [3] [4], TSIMMIS [10], Information Manifold [13], and OBSERVER [15] are some of the other related efforts. The goal of InfoQuilt is to allow users to query, analyze, reason about inter-domain relationships and analyze data available from multiple sources. However, most other systems focus only on retrieving and integrating "data" and not on the "exploring, understanding and knowledge discovery" aspects. The distinguishing features of InfoQuilt are:

- Ability to help in understanding domains and their complex relationships
- Support for functions and simulations
- Ability to model complex relationships
- Powerful semantic query interface (IScapes)

We briefly compare our vision and approach with other systems in this section. A detailed discussion appears in [17].

SIMS [3] [4] creates a model of the domain using a knowledge representation system and accepts queries in the same language. A major limitation of the system is that its mediator is specialized to a single application domain [2]. It also does not support inter-ontological relationships and functions.

OBSERVER [15] uses ontologies to describe information sources and inter-ontology relationships like synonyms, hyponyms and hypernyms across terms in different ontologies to translate a query specified using one ontology into another query that uses related ontologies. This approach of using relationships to achieve interoperability between the sources is interesting. However, it is limited to basic relationships.

TSIMMIS [10] uses a mediator-based architecture [22]. It uses Mediator Specification Language (MSL) to define mediators. The mediators are then generated automatically from these specifications. Since the MSL definitions need to be created manually, adding or removing information sources requires updating and recompiling the definitions. The mediator answers queries by relating them to pre-defined query templates. Hence, it can answer only a restricted set of queries. InfoQuilt has a dynamic planner that automatically considers newly added sources while planning IScapes.

Information Manifold [13] uses an approach similar to ours in that the user creates a world view, a collection of virtual relations and classes. The world view however does not capture semantics of the domains as InfoQuilt can using domain rules and FDs. Information sources are described to the system as a query over the relations in the world view. The user queries are also specified over relations in this world view. The sources can be specified to be either a subset of the domain or equal to a subset of the domain [14]. Hence, local completeness cannot be modeled precisely. IM uses capability records to capture query capability limitations of sources. The system arbitrarily selects a set of input parameters. This approach would not work if the source needs very specific combinations of attributes as input.

Duschka [9] present a comprehensive theory on planning queries in information integration systems. They focus on creation of maximally-contained query plans, defined as a plan that provides *all* the answers that are possible to obtain from the available sources. They show that it is necessary to consider plans with recursion in order to generate maximally-contained plans. Since a large number of information sources would be web sources that are slow relative to a source that is a database, execution of plans with recursion could be very slow. A query plan that does not necessarily return all the possible results may still be useful. We adopt this approach.

6 Conclusion

We discussed how InfoQuilt goes beyond the traditional querying techniques to provide access to and integrate data in a more semantic manner with the goal of providing a querying, analyzing and knowledge discovery environment. The key features of the system are:

- ability to model inter-domain relationships
- ability to use value-adding functions and simulations
- powerful query interface to describe complex information needs (IScapes)
- environment for knowledge discovery

We described the planning and optimization algorithms used in InfoQuilt. The planner is dynamic and flexible. It allows dynamic addition, deletion, and modification of ontologies, relationships, resources, and functions. The contributions of this paper are practical algorithms to efficiently select sources most appropriate to answer an IScape by excluding sources with redundant or no useful information in the context of the IScape, using sources in conjunction to retrieve more comprehensive information using the same set of available resources, generating executable plans that respect query capability limitations of the resources, integrating retrieved information by evaluating relationships and post-processing (evaluating functions and running simulations), and multi-threaded processing of the IScapes to exploit the parallelism in the plans.

Following are some of the planned future improvements. The Planning Agent could create backup plans that the Correlation Agent can switch to on failure. Simulations are currently supported using the framework used for functions. Simulation programs however are more complex and diverse in the kind of application (it could be an executable, a script, etc.), the method of accepting inputs, for e.g. as files from local file systems, etc. The framework currently used may not suffice as it assumes that they are available as separate functions (through wrappers if needed). Several simulations however exist as programs written using languages supported by special software, for e.g., ArcInfo, and hence, may be difficult to add to the Function Store. Different types of simulation programs therefore need to be explored more thoroughly to provide a framework better suited to support a large class of them, if not all.

References

1. Alexandria Digital Earth Prototype *http://alexandria.ucsb.edu/* 135
2. Y. Arens, C. Hsu and C. A. Knoblock. Query processing in the SIMS information mediator. In Austin Tate, editor, Advanced Planning Technology. The AAAI Press, Menlo Park, CA, 1996 146
3. J. Ambite, and C. Knoblock. Planning by rewriting: Efficiently generating high-quality plans. Proceedings of the 14th National Conference on Artificial Intelligence, Providence, RI, 1997 146
4. Y. Arens, C. A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, Vol. 6, pp. 99-130, 1996 136, 146
5. R. J. Bayardo Jr., W. Bohrer, R. Brice, et al. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In SIGMOD-97, pp. 195-206, Tucson, AZ, USA, May 1997 144
6. C. Bertram. InfoQuilt: Semantic correlation of heterogeneous distributed assets. Masters Thesis, Computer Science Dept, University of Georgia, 1998 145
7. S. Chaudhuri, R. Krishnamurthy, S. Potamianos, K. Shim. Optimizing queries with materialized views. Proceedings of international conference on Data Engineering, 1995 141
8. Clarke's Urban Growth Model, Project Gigalopolos, Department of Geography, University of California, Santa Barbara
http://www.ncgia.ucsb.edu/projects/gig/ncgia.html 146

9. O. M. Duschka. Query planning and optimization in information integration. Ph. D. Thesis, Computer Science Department, Stanford University, 1997 147
10. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, et al. The TSIMMIS approach to mediation: Data models and languages. In Proceedings of NGITS, 1995 136, 146, 147
11. M. Guntamadugu. MÉTIS: Automating metabase creation from multiple heterogeneous sources. Masters Thesis, Computer Science Department, University of Georgia, 2000 142
12. A. Y. Levy, A. O. Mendelzon, Y. Sagiv and D. Srivastava. Answering queries using views. Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of database systems, PODS-95, San Jose, CA, May 1995 141
13. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. Proceedings of the 22nd International Conference on Very Large Databases, Bombay, India, September 1996 136, 146, 147
14. A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. International Journal on Intelligent Information Systems, pp. 121-143, 1995 147
15. E. Mena, A. Illarramendi, V. Kashyap, and A. P. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. International Journal on Distributed and Parallel Databases, Vol. 8, No. 2, pp. 223-271, April 2000 136, 146, 147
16. K. Parasuraman. A multi-agent system for information brokering in infoQuilt. Masters Thesis, Computer Science Department, University of Georgia, 1998 145
17. S. Patel and A. Sheth. Planning and optimizing semantic information requests using domain modeling and resource characteristics. Technical Report, LSDIS Laboratory, University of Georgia, 2001. *http://lsdis.cs.uga.edu/proj/iq/iq_pub.html* 137, 141, 142, 144, 145, 146
18. D. Singh. An agents based architecture for query planning and cost modeling of web sources. Masters Thesis. Computer Science Department, University of Georgia, 2000 145
19. A. Sheth, Changing focus on interoperability: From system, syntax, structure to semantics. In M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman, editors, Interoperating Geographic Information Systems, Kluwer Academic Publishers, 1999 136
20. A. Sheth and W. Klas, Editors. multimedia data management - Using metadata to integrate and apply digital media. Mc Graw-Hill, 1998 136
21. S. Thacker, S. Patel, and A. Sheth. Knowledge modeling for study of domains and inter-domain relationships - A knowledge discovery paradigm. Technical Report, LSDIS Laboratory, University of Georgia, 2001. *http://lsdis.cs.uga.edu/proj/iq/iq_pub.html* 137, 140
22. G. Wiederhold. Mediators in the architecture of future information systems. IEEE Computer, 25(3), pp. 38-49 147
23. H. A. Yang, and P. A. Larson. Query transformation for PSJ queries. Proceedings of 13th International Conference on Very Large Databases VLDB-87, Brighton, England, 1987, pp. 245-254 141