Semi-Naive Query Planning For Grid Data Integration

Vasilis Samoladas

Dept. of Electronic and Computer Eng., Technical University of Crete
vsam@softnet.tuc.gr

Abstract. We consider the problem of querying multiple grid data sources under the assumption that quantitative metadata of data access costs is not available. In this case, query planning must rely on qualitative information to improve performance. We introduce a semantic approach that allows the hand-crafting of qualitative data source characterizations related to access costs. Our technique may offer substantial optimization opportunities in practical situations. In addition, it can uniformly describe data sources with reduced query capabilities, such as data-producing computations and devices.

1 Introduction

Grids promise new capabilities to scientific and bussines computing in the form of support for loosely coupled Virtual Organizations, i.e. the formation and administration of ad-hoc groups of users who can share computational and information resources safely and easily. To support such environments, applications will need data integration capabilities among sources of structured data, such as databases. A number of works already address related issues, notably data access, query execution and semantic mapping on the grid (e.g. [2, 10, 21, 17]).

Database integration technology is widely deployed today in the business sector, but less so in other data processing environments. Data integration applications on the grid may prove more challenging, if there is need for the execution of complex computations during integration. Scientific data integration for example, much more so than business data integration, is likely to operate on data computed on-the-fly, or collected by intruments, as well as data already stored in repositories. We call this *Grid data integration*.

In this paper, we view computations as implemented by Grid services, and model them as data sources of limited query capability. Succinctly, a Grid service is implemented by a number of remote procedures. Consider such a remote procedure with signature

$$\mathrm{foo}(x_1, \ldots, x_n) \rightarrow y$$

We can treat this procedure as a special type of (database) relation

$$r_{\mathrm{foo}}(x_1, \ldots, x_n, y)$$

on $n + 1$ attributes, with the additional restriction that the table cannot be retrieved in full, but only in limited ways. For example, a legal query is

$$\texttt{SELECT } y \texttt{ FROM foo WHERE } x_1 = a_1 \texttt{ AND ... AND } x_n = a_n$$

(here, a_i stand for appropriate constants). Examples of such sources include web databases [10], value-mapping services, devices and instruments, or other data integration services. Access to a service as a data source is done through an appropriate wrapper, able to translate incoming queries into procedure calls.

Our work is being driven by the vision of grid data integration in *open dynamic environments*. *Open* means that anyone is able to publish data (or services), from any location, without need for centralized administration. *Dynamic* means that data sources (or services) may be added or removed from the system in an unpredictable fashion, and with very little overhead.[1] In such environments, metadata about data sources is rarely uniformly available. A bare minimum of metadata is needed for semantic integration, and is usually offered in the form of statements over some well-known ontology, which we will call the *global ontology* (it is sometimes called the global scema, or the federated schema). Such metadata is needed in order to translate queries over the global ontology into actual query plans over the available data sources.

Data sources often shun the complexity of offering access-cost related metadata; for example, the OGSA-DAI [2] middleware does not yet provide support for this. In these situations, *quantitative* (cost-based) query planning over such data sources is not an option. Open and dynamic environments pose additional challenges to the provision of cost-related metadata, since:

- Typical cost models require cost-related information to be *uniformly* available, i.e. over all data sources.
- For those data sources that are services (text extractors, image processors, remote sensors, etc.) a well-understood access cost model may not be available.
- Even if data sources are relational databases, quantitative cost estimates may be difficult to derive, if, for example, the data to be retrieved is computed by a complex query, e.g. a local multi-join query with aggregates.
- Network costs may be highly volatile in wide-area Grids.

There are techniques to address the above challenges, at least for business data integration; they are reviewed in brief later.

Our technique. We briefly describe our approach to qualitative query planning. For the sake of simplicity, in the following we adopt the relational data model with its standard notation, although out techniques can be adapted to other data models as well.

The main tool in our technique is a new type of semantic assertion: *virtual access paths (VAPs)*. Let r be a relation, $X = \{A_1, \ldots, A_n\}$ and $Y = \{B_1, \ldots, B_m\}$

[1] A most familiar open dynamic environment is a Peer-to-peer network. However, we do not study peer-to-peer networking and search issues in this paper.

be sets of attribute names of r, and $\tau \in \mathcal{T}$ be *statement* (for now treat τ as just a label). Syntactically, a VAP takes the form $X \xrightarrow{\tau} Y$. Informally, the meaning of this assertion is the following:

> For all $a_1, \ldots, a_n$, statement τ holds for the execution of query
>
> $$\pi_{B_1 \ldots B_m}\left(\sigma_{A_1=a_1 \wedge \cdots \wedge A_n=a_n}(r)\right)$$
>
> (where π denotes the relational projection and σ the relational selection operator).

Statement τ is a qualitative statement about the cost of the above operation.

For example, assume that the relation r(K,A,B,C) is stored in a DBMS and there exists an index (say, a B-tree) on attribute K. We could then assert the following:

$$\{\mathrm{K}\} \xrightarrow{\mathrm{ind}} \{\mathrm{K}, \mathrm{A}, \mathrm{B}, \mathrm{C}\}$$

An optimizer can use this assertion to take advantage of the index in planning for a query that will access r. Thus, in a sense, VAPs can be seen as statements on the existence of access paths in a data source. Of course, actual access paths corresponding to a VAP may not physically exist; the assertion may simply describe a data source's access behaviour. A desirable aspect of VAPs is that they can often be asserted from real-world knowledge which has an impact on the statistics of attribute values. For example, any given person is parent to only a few children, there are comparatively few people with a given first and last name (even if it is "John Smith"), and so forth. We can assist query planning with VAPs asserting these facts, without providing quantitative estimates.

A number of desirable properties make VAPs well suited for query planning in open, dynamic environments:

- VAPs can easily be composed by a human, and are human-readable. Thus, users can easily annotate data sources which do not provide adequate meta-data, and also can easily detect errorneous data source annotations.
- Data sources may offer very restricted query capabilities, thus limiting the allowed queries. VAPs can describe succinctly the set of viable queries to such sources.
- Reasoning over VAPs can cope with missing information in a natural way.
- VAPs can be combined with cost-based optimization techniques.

This papers is organized as follows: Section 2 contains a more thorough discussion of related work. Section 3 defines VAPs and the necessary inference rules to apply them in query planning. Section 4 discusses a particular architecture for query planning and execution, based on VAPs. Finally, section 5 discusses a number of remaining issues, as well as our experience with an initial implementation of a VAP-based query engine for Infosleuth [5].

2 Related work

Early data support on the Grid focused in the management of file-based data, mainly to cater scientific applications. Progressively, the adoption of the Grid by business applications, as well as the management needs of the Grid itself, have brought the use of data bases, mainly relational ones, at center stage. The need for interoperable, transparent access to different database systems is recognized in a number of projects (e.g. DataGRID, Globus, OGSA-DAIS) as well as in recent releases of commercial products (e.g. Oracle, DB2, MySQL). The OGSA-DAI architecture, recently incorporated into the Globus toolkit, provides service-based access and integration to relational and XML data sources, in cooperation with Grid security and resource discovery services.

Besides access to individual databases, Grid tools for data integration are in development. A relevant work of interest is OGSA-DQP [17], which implements a distributed query service based on OGSA-DAI. A nice feature of OGSA-DQP is the ability to decompose and schedule a query execution plan on different processing nodes on the Grid.

Data integration is a well-studied subject, with a number of innovative systems already implemented (e.g. Datajoiner [20], Garlic [7], Tukwila [9], InfoSleuth [13], DISCO [18], TSIMMIS [6], Information Manifold [11]) which deal with heterogeneous data sources and models, and the problems of integration thereof. In addition, there is a great number of techniques developed by researchers independently of any actual system. The emphasis of this work has been on the semantic integration problem.

Query planning has also been adequately addressed, but mainly under an operational assumption of a static or slowly changing collection of data sources, where additional effort in the crafting of mediators that can capture the behaviour of individual sources is justified. Several techniques have been proposed. Roughly, they fall into two categories.

1. A number of techniques (e.g. [23, 8, 16]) attempt to construct sophisticated statistical models of the behaviour of data sources, in both static and dynamic environments. Models are often constructed automatically, by executing statistically significant numbers of sample queries across the data source. This approach requires substantial preprosessing costs before a data source is modelled adequately. Furthermore, the network's behaviour is not separated from the overall behaviour, thus a data source's cost model constructed by one user may not be correct for other users.
2. Another family of techniques cope with the absense of cost-related metadata by avoiding the problem. They propose, on a per-query basis, the adaptation of the query plan on-the-fly, as query evaluation proceeds and the actual behaviour of data sources is being observed [3, 19, 9]. These techniques benefit greatly from even rough or imprecise information to initialize query execution. In this sense, they are compatible and can benefit from our work.

There are also several semantics-based query planning techniques [8, 22, 11] for data integration. The approach in these works however relates to the use of

integrity constraints for logical query reformulation, whereas our VAPs characterize performance of physical data operations. For example, Hsu and Knoblock [8] use semantic constraints such as primary and foreign keys, to simplify distributed queries (e.g. remove unnecessary joins). A notable semantic approach to operations is introduced by Zhu et al. [24], where qualitative variables are used to model dynamic aspects of the execution environment in multidatabase systems.

Another related area is the emerging paradigm of peer-to-peer databases. The P2P architecture is well suited for ad hoc collaborations in the framework of grid-based virtual organizations, which may need to accomodate data integration accross hundreds of data sources. This is a very interesting application scenario of grid technology and is currently under intensive research (see [1] for an extensive list of ongoing projects). The techniques of this paper are very appropriate in this setting.

3 Virtual Access Paths

We will now define Virtual Access Paths in some detail, describe the basic reasoning operations on them and outline the basic usage for describing data sources. In what follows, we will use standard relational database theory notation. Letters $A, B, C, \ldots$ stand for attribute names, and letters X, Y, Z, W stand for sets of attribute names. Relations are represented by $r, s, \ldots$. We will use the selection ($\sigma_p(r)$), projection ($\pi_X(r)$), join ($r \bowtie_p s$), semi-join ($r \ltimes_p s$) and union ($r \cup s$) relational operators to form relational expressions. Also, by $[r]$ we denote the set of attributes (a.k.a. the scheme) of relation r.

3.1 Formal definition

Let $(\mathcal{T}, \rightarrowtail)$ be a finite partially ordered set, where $\mathcal{T}$ is a set of *statements* and $\rightarrowtail \subseteq \mathcal{T}^2$ is a reflexive, antisymmetric and transitive relation over $\mathcal{T}$. Also, let $\mathcal{A}$ be a finite set of *attribute labels*. A Virtual Access Path is denoted as

$$X \xrightarrow{\tau} Y$$

where $X, Y \subseteq \mathcal{A}$ and $\tau \in \mathcal{T}$. VAPs satisfy the following axioms (for all $X, Y, Z, W \subseteq \mathcal{A}$ and $\tau, \tau' \in \mathcal{T}$):

$$X \xrightarrow{\tau} X \tag{1}$$

$$X \xrightarrow{\tau} Y \Rightarrow X \cup Z \xrightarrow{\tau} Y \tag{2}$$

$$X \xrightarrow{\tau} Y \wedge Y \cup Z \xrightarrow{\tau} W \Rightarrow X \cup Z \xrightarrow{\tau} W \tag{3}$$

$$\tau \rightarrowtail \tau' \wedge X \xrightarrow{\tau} Y \Rightarrow X \xrightarrow{\tau'} Y \tag{4}$$

Note that Eqs. 1–3 are well known as *Armstrong's axioms*, and form a sound and complete axiom set for *functional dependencies*. Also, Eq. 4 is the only rule pertaining to $\mathcal{T}$.

Let $\mathcal{F}$ be a set of VAPs. $\mathcal{F}$ *entails* a VAP $X \xrightarrow{\tau} Y$, iff the latter can be deduced by $\mathcal{F}$ and the above axioms. By the closure $\mathcal{F}^*$ of set $\mathcal{F}$, we denote the set of all VAPs entailed by $\mathcal{F}$.

Instead of reasoning directly over the axioms, entailment can be computed very efficiently by well-known linear-time algorithms (which we omit). We use the following notation: for $\mathcal{F}$ a set of VAPs, define $\mathcal{F}(X, \tau)$ to be the maximum set of attributes such that $\mathcal{F}$ entails $X \xrightarrow{\tau} \mathcal{F}(X, \tau)$.

3.2 Statements

VAPs are qualitative statements describing dependencies between sets of attributes. However, *the dependencies must conform to axioms 1–4*. Such dependencies include functional dependencies (a well-known type of semantic contraint). We are interested in types of dependencies that assert access-related properties of data sources. Each type of dependency will be associated with a *statement* τ. We present four useful statements below, by way of example.

Indexes. Let r be a relation stored in a DBMS. Existence of a B-tree index on r, say over attribute A, *strongly suggests* that the query $\sigma_{A=c}(r)$:

- will be computed very efficiently, and
- the result of the query is small (A is selective).

We can express the existence of the index on relation r as a VAP assertion:

$$A \xrightarrow{\text{ind}} [r].$$

where ind $\in \mathcal{T}$ is a statement. More generally, we should characterize any efficient and selective access path using ind statements.

Selectiveness. We say that an attribute A is *selective* for a relation r, if the size of the queries

$$\sigma_{A=a}(r)$$

is small on average, (a is a constant). We could express this characterization as a VAP:

$$A \xrightarrow{\text{sel}} [r].$$

This statement is weaker than $A \xrightarrow{\text{ind}} [r]$, because computing the selection need not be efficient (although the result is small). Thus, we have

$$\text{ind} \rightarrowtail \text{sel},$$

that is, ind implies sel.

Functional dependency. A stronger form of selectivenes is functional dependency. Functional dependencies are semantic constraints that usually describe relation keys. In RDBMS they are enforced by unique indices. Thus, we are justified to have

$$\text{fd} \rightarrowtail \tau.$$

Accessibility. As discussed, some data sources provide restricted access to the data. For example, an internet search engine (such as Altavista) or a name service (like DNS), will only support specific queries. VAPs can be used to model certain types of restrictions of this sort. For example, assume that a data source only admits queries for a relation r which select on some attribute, A. This is expressed as a VAP by

$$A \stackrel{\text{acc}}{\rightarrow} [r].$$

Of course, unrestricted relations can be described by

$$\emptyset \stackrel{\text{acc}}{\rightarrow} [r].$$

Accessibility is a very weak statement made of a data source, so we might as well consider it the weakest in $\mathcal{T}$:

$$\tau \rightarrowtail \text{acc}.$$

Other examples. We present additional VAP-based characterizations of typical relations and data sources, for the statement set described above:

$$\mathcal{T} = \{\text{fd}, \text{ind}, \text{sel}, \text{acc}\}$$

with $\text{fd} \rightarrowtail \text{ind} \rightarrowtail \text{sel} \rightarrowtail \text{acc}$.

- A singleton RDBMS relation r:

$$\emptyset \stackrel{\text{fd}}{\rightarrow} [r].$$

- A very small RDBMS relation:

$$\emptyset \stackrel{\text{ind}}{\rightarrow} [r].$$

- A large RDBMS relation without indices:

$$\emptyset \stackrel{\text{acc}}{\rightarrow} [r].$$

- A large RDBMS relation with an index on attribute K and an attribute A taking a small nuber of (e.g. A might be of type boolean):

$$\emptyset \stackrel{\text{acc}}{\rightarrow} [r], \quad K \stackrel{\text{ind}}{\rightarrow} [r], \quad \emptyset \stackrel{\text{sel}}{\rightarrow} A.$$

- A remote procedure $f(A_1, \ldots, A_n)$ that returns a tuple of the form $(B_1, \ldots, B_m)$. This can be modelled as a relation on the schema $(A_1, \ldots, A_n, B_1, \ldots, B_m)$ and characterized by

$$\{A_1, \ldots, A_n\} \stackrel{\text{fd}}{\rightarrow} \{B_1, \ldots, B_m\}.$$

- A computed relation (view) defined by the following SQL query

```
SELECT A, sum(B) AS B
FROM R
GROUP BY A
```

can be characterized by:

$$A \stackrel{\text{sel}}{\rightarrow} \{A, B\}, \;\; \emptyset \stackrel{\text{acc}}{\rightarrow} \{A, B\}.$$

Note that in this case, $A \stackrel{\text{fd}}{\rightarrow} \{A, B\}$ would not be appropriate, since we defined fd above to imply efficiency. Here, without an index on A, efficiency cannot be assumed.

3.3 VAPs for relational expressions

We now discuss how VAPs of relational expressions can be derived from VAPs of the expression operands.

The select operator Without loss of generality, we consider selection operators whose predicate consists of a single equality of the form $A = a$ or $A = B$ where A and B are attributes and a a constant. Conjunctions and disjunctions (but not negations) of such equalities can be handled by the following two rules:

$$\sigma_{p \wedge q}(r) = \sigma_p\left(\sigma_q(r)\right)$$

$$\sigma_{p \vee q}(r) = \sigma_p(r) \cup \sigma_q(r)$$

Given a set $\mathcal{F}$ of VAPs for r, the set of VAPs for $\sigma_{A=c}(r)$ is obtained by adding $\emptyset \stackrel{\text{fd}}{\rightarrow} A$ to $\mathcal{F}$, and the set of VAPs for $\sigma_{A=B}(r)$ is computed by adding $A \stackrel{\text{fd}}{\rightarrow} B$ and $B \stackrel{\text{fd}}{\rightarrow} A$ to $\mathcal{F}$.

For atomic predicates other than equalities, we may leave the set $\mathcal{F}$ unchanged. Alterantively we may try to determine the selectivenes of the predicate, and add appropriate VAPs to $\mathcal{F}$.

The join operator Let the multi-way join be over relations $r_1, \ldots, r_n$ and p be the join predicate. Let $\mathcal{F}_i$ be the set of VAPs for operand r_i. We compute the VAPs for the join by taking $\mathcal{F}$ to be the union of all $\mathcal{F}_i$, where we take care to rename the attributes in all VAPs in each $\mathcal{F}_i$ to unique names. $\mathcal{F}$ is the set of assertions applying to the cartesian product $r_1 \times \ldots \times r_n$. We process the predicate p on $\mathcal{F}$ as for the select operator. This is justified by the well-known relational theorem:

$$\underset{p}{\bowtie}(r_1, \ldots, r_n) \;=\; \sigma_p(r_1 \times \ldots \times r_n)$$

As in the case of the select predicate, the join predicate is assumed to consist of equalities combined by $\wedge$ and $\vee$.

The union operator Let $\mathcal{F}_1$ and $\mathcal{F}_2$ represent VAP sets for r_1 and r_2, where r_1 and r_2 have the same schema. Let $\mathcal{F}$ be a set of VAPs for the relation $r_1 \cup r_2$.

In general, whether a VAP $X \xrightarrow{\tau} Y$ belongs to $\mathcal{F}^*$ is determined by whether it belongs to $\mathcal{F}_1$ and/or $\mathcal{F}_2$. For most types of statements τ, a VAP $v \equiv X \xrightarrow{\tau} Y \in \mathcal{F}^*$ only if $v \in \mathcal{F}_1^* \cap \mathcal{F}_2^*$. However, the converse does not always hold; for example, for $\tau = \text{fd}$, $X \xrightarrow{\text{fd}} Y$ may hold in both r_1 and r_2 but not in their union (of course, in this example $X \xrightarrow{\text{ind}} Y$ will be inferred for the union).

The most general way of computing $\mathcal{F}$ is to compute the closures $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ and from them compute $\mathcal{F}$. This is a rather naive and potentially very expensive way. A number of techniques can be used to speed up the computation, but we will not discuss them in this paper.

The project operator Let $\mathcal{F}$ be the VAP set of relation r. The VAP set $\mathcal{F}_\pi$ of $\pi_X(r)$ should entail exactly those VAPs $Y \xrightarrow{\tau} Z \in \mathcal{F}^*$, such that $Y \subseteq X$ and $Z \subseteq X$. This is easy to see if we remember that the intuitive meaning of VAP $Y \xrightarrow{\tau} Z$ over r is that "τ holds for $\pi_Z(\sigma_{Y=y}(r))$", and we also note that for $Y, Z \subseteq X$, and any relation r,

$$\pi_Z(\sigma_{Y=y}(r)) = \pi_Z(\sigma_{Y=y}(\ \pi_X(r)\ ))$$

Like in the case of union, the inference for the project operator can also be quite expensive. In fact, it is possible to contrive cases where the size of $\mathcal{F}_\pi$ must be exponetial to the size of $\mathcal{F}$ (even if $\mathcal{T}$ is a singleton set). However, as in the case of the union, such cases do not occur in practice. We will again omit the description of efficient computation techniques.

4 Query planning with VAPs

So far we have outlined the characterization of data sources using VAPs, and we have also seen how to derive VAPs for relational expressions from VAP-based characterizations of the operands. However, it may not be clear how these characterizations can be used for query planning. VAPs by themselves offer a rather crude characterization of a data source. Yet, in the absense of more detailed cost-related metadata, VAPs can assist in selecting more efficient query plans. Altough they can be used in more than one way to this end, we will make the discussion more concrete by focusing on a particular planning issue; that of augmenting a query plan using semijoin reduction.

4.1 Semijoin reduction

Reducing distributed queries. Consider a (natural) join query between relations $r(AB)$ and $s(BC)$, residing on different data sources. Assume that all processing occurs on a single site, but the data sources can execute local queries. Planning the execution of this query could produce a plan similar to the following:

1. Copy r to the processing site.
2. Copy s to the processing site.
3. Compute $r \bowtie s$ by an appropriate join algorithm.

When a planner accepts such a query from the user, in the absense of more information the above plan seems the most appropriate alternative. However, if it is known that r is a very small relation (say, contains only a few tuples), then a much better approach would be the following:

1. Copy r to the processing site.
2. Let $b_1, \ldots, b_n$ be the different values in $\pi_B(r)$.
3. Retrieve $u = \sigma_{B=b_1 \vee \cdots \vee B=b_n}(s)$.
4. Compute $r \bowtie u$.

In classic relational terminology, we say that s is reduced to u by r, or, in relational algebra, $u = s \ltimes \pi_B(r)$.

This query plan can be significantly faster than the previous one, if n is small. In fact, values of n up to a few thousands per single query can easily be supported by modern RDBMS. Of course, for even larger n, multiple queries can be used.

Limited-capability data sources. Another application of semijoin reduction arises from the integration of data sources with limited query capabilities. By way of example, assume that a relation $h(H)$ is a (single-attribute) table of host names, and that a DNS service (say, RPC-based) is modelled as a relation $d(H, A)$ of (hostname, ip-address) pairs. Translation of the names in h to IP addresses can be expressed as query $h \bowtie d$, where it should be understood that h is the outer relation. The DNS service wrapper (which provides a data-source API for the service) is then in a position to invoke the service appropriately.

4.2 Semijoin planning with VAPs

We will examine semijoin planning based on VAPs, under some simplifying assumptions about query execution. We are roughly following the model of execution of Infosleuth [4].

We consider select-project-join queries over a global relational ontology, consisting of relations, each relation possibly partitioned over multiple data sources. A query is mapped on the relevant data sources, and, for each data source, one or more *local queries* are formed. Local queries try to push as many operations as possible (such as selections and joins) to each data source. Of course, some data sources may not support such capabilities.

The results of the local queries are in turn inputs to a *global query* Q. This query will have the form

$$Q = \underset{p}{\bowtie}(e_1, e_2, \ldots, e_k)$$

where e_i is a (possibly singleton) union of local queries. Our task is then to reformulate the top-level join operation of the global query with semijoin reduction.

Let relation r be associated with a set of VAPs $\mathcal{F}$. We say r is τ-reduced (with respect to $\mathcal{F}$), iff $\mathcal{F}(\emptyset, \tau) = [r]$ (in other words, iff $\mathcal{F}$ entails $\emptyset \xrightarrow{\tau} [r]$). Let $\mathcal{F}_Q$ be the set of VAPs computed from the VAP characterizations of the data sources, as described previously. Also, let $\mathcal{F}_i$ stand for the set of VAPs associated with e_i.

Under-specified queries. One question of interest is whether a query over data sources with limited capabilities is adequately specified. Assuming that we used VAPs over statement acc (§3.2) to model capabilities of data sources, we have the following:

Theorem 1. *A query Q is adequately specified iff it is acc-reduced.*

Proof. We omit the proof. □

Note that showing a query adequately specified is not the same as planning for the query. Essentially, to find a viable plan for evaluating Q, we must determine a reducer ρ_i for each operand e_i, i.e. a set of operands ($\rho_i \subseteq \{e_1, \ldots, e_n\}$), such that the expression

$$u_i = e_i \underset{p}{\ltimes} \Big(\underset{e \in \rho_i}{\times} e \Big)$$

is acc-reduced.

We can then determine the set of viable plans via a closure computation, where in each step we use those operands that are already acc-reduced to form reducers for more operands. Essentially, the closure computation mirrors the closure-like proof of acc-reducedness of Q. We omit the details because of space restrictions. Note however that this approach will in general produce a set of possible plans. Selecting among them can be based on other considerations besides viability, such as performance.

Semijoin reduction. We are also interested in classic semijoin reduction, as it can potentially reduce the costs of query evaluation by orders of magnitude. Obviously, the choice of reducers will have an impact on performance. As above, we can enumerate a number of semijoin query plans, using closure-like iteration as for the case of the previous paragraphs. Again, details are omitted.

5 Discussion

5.1 VAP-based query planning in Infosleuth

A prototype of a VAP-based query planner was developed in 1999-2000, within the InfoSleuth Project at MCC [13, 14, 12], and tested against a real data-integration application, the Environmental Data Exchange Network (EDEN) [15, 5]. EDEN queries access large databases located in Idaho, Georgia, Maryland, Tennessee, Texas and Virginia, and in Europe. The highly distributed nature of the application, coupled with the large volumes of data involved, made semijoin query processing essential. Insufficiently constrained queries to remote data

sources had unacceptably slow response times. Thus, EDEN provided a good vehicle for testing the validity and effectiveness of our approach.

Even simple queries in EDEN would typically join large, horizontally partitioned relations. Relations in the (federated) EDEN schema were defined as high-arity join views. Combined with the large number of participating data sources, even a simple 2-way join SQL query (on the federated schema) could easily result in a query plan accessing more than 30 (physical) relations, combined through joins and unions. Thus, we were able to test our techniques on some very large examples. Different EDEN databases had very different response times. In addition, response times of data sources were highly variant, both in the short term (e.g. depending on the time of day) and in the long term (over several days).

We implemented our technique in a first-solution query engine, and ran a number of typical queries. Simply employing semi-join reduction allowed queries that were infeasibly slow without it. Other queries, feasible even without semi-join processing, improved both in response time and completion time using our approach, typically by an order of magnitude. Planning times were kept sub-second even for the larger queries.

Despite the large EDEN schema, a colleague of ours composed and debugged *by hand* the VAPs for the whole EDEN application in less than 1 hour. This is very encouraging evidence that VAPs are an intuitive method of characterising the access behaviour of data sources.

5.2 Current work

We are in the process of implementing a VAP-based query engine, based on the OGSA-DAI facility that as of recently is part of the Globus toolkit. We are particularly interested in extending our techniques to query planning issues beyond semi-join reduction. One interesting direction is the planning of grid-distributed query execution, along the lines of OGSA-DQP, in particular the extent to which VAPs can influence query parallelization decisions on Grid resources.

References

[1] P2p model of computing and databases: a bibliography. `http://www.cs.toronto.edu/~kiringai/p2p-db.html`.

[2] M. Antonioletti et al. OGSA-DAI: Two years on. In *Future of Grid Data Environments, GGF-10*, 2003.

[3] Ron Avnur and Joseph M. Hellerstein. Eddies: continuously adaptive query processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 261–272, 2000.

[4] R. Bayardo et al. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 195–206. ACM Press, Jun 1997.

[5] Jerry Fowler, Brad Perry, Marian H. Nodine, and Bruce Bargmeyer. Agent-based semantic interoperability in infosleuth. *SIGMOD Record*, 28(1):60–67, 1999.

[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[7] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proceedings of the International Conference on Very Large Databases*, 1997.

[8] Chun-Nan Hsu and Craig A. Knoblock. Semantic query optimization for query plans of heterogeneous multidatabase systems. *Knowledge and Data Engineering*, 12(6):959–978, 2000.

[9] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1999.

[10] I. Kozima and S. M. Pahlevi. Design and implementation of OGSA-WebDB - a service based system for making existing web databases grid-ready. In *Future of Grid Data Environments, GGF-10*, 2003.

[11] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 251–262, Bombay, India, 1996. VLDB Endowment, Saratoga, Calif.

[12] M. Nodine, W. Bohrer, and A. Ngu. Semantic multibrokering over dynamic heterogeneous data sources in InfoSleuth. In *Proceedings of the International Conference on Data Engineering*, 1999.

[13] Marian Nodine, Jerry Fowler, Tomasz Ksiezyk, Brad Perry, Malcolm Taylor, and Amy Unruh. Active information gathering in InfoSleuth. *International Journal of Cooperative Information Systems*, 9(1/2), 2000.

[14] B. Perry, M. Taylor, and A. Unruh. Information aggregation and agent interaction patterns in InfoSleuth. In *Proceedings of the International Conference on Cooperative Information Systems*, 1999.

[15] G. Pitts and J. Fowler. Collaboration and knowledge sharing of environmental information: The EDEN project. In *Proceedings of the IEEE International Symposium on Electronics and the Environment*, May 1998.

[16] M. Roth, F. Ozcan, and L. Haas. Cost models DO matter: Providing cost information for diverse data sources in a federated system. In *Proceedings of the International Conference on Very Large Databases*, 1999.

[17] J. Smith, A. Gounaris, P. Watson, N. W. Paton, A. A. Fernandes, and R. Sakellariou. Distributed query processing on the grid. In *Proc. Grid Computing 2002*, pages 279–290. Springer, LNCS 2536, 2002.

[18] A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to heterogeneous data sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, 10(5), 1998.

[19] T. Urhan, M. Franklin, and L. Amsaleg. Cost based query scrambling for initial delays. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1998.

[20] S. Venkataraman and T. Zhang. Heterogeneous database query optimization in DB2 Universal DataJoiner. In *Proceedings of the International Conference on Very Large Databases*, 1998.

[21] T. Walsh, S. Karimi, K. Gamiel, J. Morris, and L. Ramakrishnan. Collection Manager: Integrating diverse data sources on the grid. In *Future of Grid Data Environments, GGF-10*, 2003.

[22] D. Woelk, P. Cannata, M. Huhns, W. Shen, and C. Tomlinson. Using Carnot for enterprise information integration. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, 1993.

[23] Qiang Zhu and Per-Ake Larson. A query sampling method of estimating local cost parameters in a multidatabase system. In *ICDE*, pages 144–153, 1994.

[24] Qiang Zhu, Yu Sun, and S. Motheramgari. Developing cost models with qualitative variables for dynamic multidatabase environments. In *Proc. of Intl. Conf. on Data Engineering (ICDE)*, pages 413–424, 2000.