An Experience Report on Designing and Building OGSA-DQP: A Service Based Distributed Query Processor for the Grid

M. Nedim Alpdemir[1], Arijit Mukherjee[2], Anastasios Gounaris[1],
Alvaro A. A. Fernandes[1], Norman W. Paton[1], Paul Watson[2]

Department of Computer Science [1]
University of Manchester
Oxford Road, Manchester M13 9PL
United Kingdom

Department of Computing [2]
University of Newcastle upon Tyne
Newcastle upon Tyne NE1 7RU
United Kingdom

5th September 2003

Abstract

OGSA-DQP is a distributed query processor exposed to users as an OGSA-compliant Grid service. This service supports the compilation and evaluation of queries that combine data obtained from multiple services on the Grid, including Grid Database Services (GDSs) provided by OGSA-DAI project. Not only does OGSA-DQP support integrated access to multiple Grid services, it is itself implemented as a collection of interacting Grid services. Thus, OGSA-DQP is an example of a Grid service with a significant amount of aggregated functionality that, in addition, illustrates how Grid service orchestrations can be used to perform reasonably complex, data-intensive parallel computations. OGSA-DQP prototype has been released and it is downloadable from www.ogsadai.org.uk/dqp/. This paper contributes an experience report on the design of OGSA-DQP, with a particular emphasis on the issues, challenges and trade-offs that the OGSA-DQP development team have had to contend with.

1 Introduction

Both commercial and scientific applications are increasingly conceived of as distributed computations. Grid technologies have been developed with a view to easing the efficient sharing of resources within a heterogeneous distributed environment, and the Open Grid Services Architecture/Infrastructure (OGSA/OGSI) [5, 13] is now emerging as the standard approach to a service-oriented view of Grid computing. Taken together, these developments have highlighted the need for middleware that provide developers of user-level functionalities with a more abstract view of Grid technologies. One of the most important, and pervasive, areas in which this need is acutely felt is that of data access and integration.

From its inception, Grid computing has provided mechanisms for data access that, from an application developer's perspective, lie at a much lower level than those provided by commercial database technology. However, the data management problems in Grid computing are not likely to be less complex, rather the contrary, insofar as in all relevant aspects (viz., data volumes, structural complexity and semantic content) data in the Grid is likely to be at least as complex as that found in current commercial environments. Furthermore, in those applications for which Grid solutions seems particularly appropriate (e.g., scientific ones), data is often more fragmented and more in need of computationally-demanding analyses than in classical Web applications (e.g., e-commerce ones). Thus, high-level data access and integration services are needed if applications that have large amounts of data with complex structure and complex semantics are to benefit from the Grid.

Standards for data access are emerging [2], and middleware products that are reference implementations of such standards are already available [4]. Distributed query processing technology is one approach to meeting the need for high-level data access and integration services given the availability of such standards and software.

OGSA-DQP [1] is a distributed query processor designed with a view to addressing the concerns above. To application developers it can be viewed simply as an OGSA-compliant Grid service. Behind the scenes, the query execution plans generated by OGSA-DQP are themselves service orchestrations. Thus, the design and development of OGSA-DQP provides some insight into some of the issues, challenges and trade-offs faced by designers and developers of Grid services.

This paper contributes an experience report on the design and devlopment of OGSA-DQP. It is structured as follows. Section 2 describes the motivation behind the development of OGSA-DQP as well as the goals that were aimed at, the issues arising and the challenges faced. Section 3 provides some information on technologies that OGSA-DQP builds upon and extends. Section 4 briefly recalls earlier approaches in distributed data management and points to ongoing work that shares goals with OGSA-DQP or responds to similar challenges. Section 5 starts by placing OGSA-DQP in relation to the approaches and the ongoing work described in Section 4, and proceeds to describe the main characteristics of OGSA-DQP and the design choices that led to them. Section 6 reflects on the consequences of the design choices and attempts to draw some lessons of general value to the budding community of Grid service developers. Section 7 draws some conclusions.

2 Motivation, Goals, Issues and Challenges

The development of OGSA-DQP is motivated by the related aims of responding to application needs and benefitting from the availability of computing infrastructure of increasing sophistication.

There is, therefore, a pull by applications in areas (both commercial and scientific) where overwhelming amounts of structurally complex data constitute a potentially important, but typically underexplored, resource. Such data tends to lie stored in very diverse, structurally dissimilar, isolated data sources. Thus, before one can analyse the data, one must have overcome impediments arising from heterogeneity in data resources, and these range from comparatively simple connectivity issues to more challenging integration concerns.

There is, in addition, a push from context and infrastructure. The service-oriented view of Grid abstractions and protocols holds the promise of not just dynamic resource discovery but also dynamic resource allocation and utilization. This presents an opportunity (and many challenges) to middleware developers insofar as one can now aim at designing and implementing value-adding services of increasing complexity in a cost-effective manner by dynamically discovery, allocating, using, and releasing high-quality (and hence potentially costly) computational and data resources.

In response to these motivations, the design and implementation of OGSA-DQP has been guided by goals that include: to expose as a Grid service an integrated approach to accessing and analysing data; and to provide means for declarative orchestration of integrated retrieval and analysis of data from independent sources, which suggests the need to exploit Grid abstractions for on-demand allocation of resources required for a task; to provide transparent, implicit support for parallelism and distribution; and to provide homogeneous access to heterogeneous data sources.

Some important design issues arising from the goals above seem to have a general interest. The overarching issue is that of mapping distributed query processing concepts and techniques onto a service-based infrastructure. In most respects, the crucial task is that of re-engineering a query processor engine in a service setting. Thus, there is a need to consider how OGSA-DQP should be presented to applications, and what functionalities it should expose. In addition, there is a need to consider how OGSA-DQP components should be internally realized. More specifically, the design needs to address the questions of how functionalities and responsibilities should be partitioned among OGSA-DQP components, what interaction discipline to impose on OGSA-DQP components, which control and which data flows to establish among components, and what component lifetime management policy to adopt.

Given that OGSA-DQP is essentially a high-throughput distributed data-flow engine that assumes (as explained in Section 4) that data resources are visible through homogeneous interfaces, the following are some of the main challenges arising for designers and implementors: to automate complex, onerous, expert configuration decisions on behalf of users; to take decisions dynamically, on an as-needed basis; to obtain the right grain in partitioning and scheduling of computation loads; to keep both interoperation overheads and the need for context switches low; to achieve efficient throughput in data movements; and, to allocate resources lazily, and to free them up as early as possible.

3 Background

The service-based system described here (i.e, OGSA-DQP) re-uses components of and builds on experience gained from our previous work on parallel and distributed query processing. Polar [12] is a parallel query processing engine which subsequently evolved into Polar* [11] a Grid-enabled, distributed system using MPICH-G. OGSA-DQP re-uses most of the optimisation, partitioning and scheduling capabilities of the Polar* engine and re-implements the query execution engine in a service setting.

OGSA-DQP uses the infrastructure provided by the reference implementation of OGSA/OGSI, viz., Globus Toolkit 3 (GT3) [9], which implements a service-based architecture over virtualised resources referred to as Grid Services (GSs). OGSA-DQP also builds upon the reference implementation of the GGF Database Access and Integration Services (DAIS) standard, viz., OGSA-DAI [4]. This is a framework built on top of GT3 GSs and extending the latter with Grid Data Services (GDSs). GDSs insulate users from certain aspects of data source heterogeneity, ensure that metadata and data held in a particular data source are accessed via a standard, well-defined and uniform interface, and use OGSA/OGSI infrastructure services for mechanisms such as data transport and authentication. The remainder of the paper assumes familiarity with OGSA and OGSA-DAI.

4 Related Work

Two general approaches to the problem of providing high-level data access and integration services over heterogeneous, autonomous, distributed data resources have been used, viz., the federation/distribution of database management systems (DBMSs) and the use of mediator/wrapper middleware.

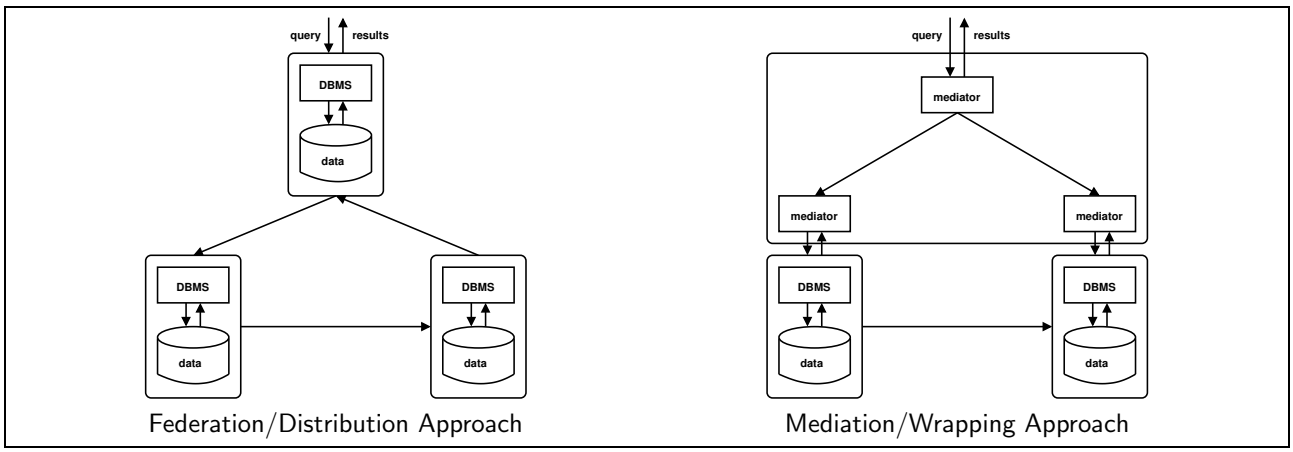


Figure 1: Approaches to Distributed Data Management

The federation approach [10] is illustrated to the left of Figure 1. Note that the grain of components is at the level of DBMSs. This makes it complex to manage because, at every node, there always is an autonomous DBMS. Resource allocation is static (i.e., the federation is a rather rigid configuration) and optimisation cannot take advantage of evolving circumstances in the execution environment (e.g., the freeing-up of extra computational resources).

An alternative approach, allowing for finer-grained components, is to use mediator/wrapper middleware [8], as illustrated to the right of Figure 1. The autonomy of DBMSs is now made less of an issue by the wrappers used to reconcile differences and project out a global schema for users to query against. Note, however, that often there is only one mediator in one fixed location, which is an obstacle to the exploitation of evolving characteristics of dynamic environments.

There are a number of initiatives which offer solutions that can be seen as examples of federation or mediator/wrapper approaches described above. In particular, the following appear to be addressing similar concerns to that of OGSA-DQP. Therefore we will compare and contrast their approaches to challenges posed by the problem with those of OGSA-DQP.

Garlic/Discovery Link [6] is a wrapper/mediator-based approach to data resource integration. Garlic is similar to OGSA-DQP in that it adopts a mediator approach to handle query optimisation and execution over distributed data sources, whose heterogeneity is hidden behind wrappers. While OGSA-DQP uses OGSA-DAI wrappers, and hence conforms to the GGF DAIS standard, Garlic wrappers are specific to it. Another difference is that the query evaluation capabilities provided by OGSA-DQP are inherently more powerful in that it uses an algebra with implicit parallellism, while Garlic does not. In contrast, the Garlic optimiser benefits from custom-built wrappers and delivers more value in terms of schema integration. Note, however, that since OGSA-DQP relies on GDSs for data retrieval it would not be difficult (from an architectural point of view) for OGSA-DQP to make use of Grid Services that perform schema integration of GDSs.

SkyQuery [7] is a wrapper-mediator system that uses a web-services approach in its middleware layer. SkyQuery is very specialised: it supports a specific type of a spatial query, called the cross match query, over federated data

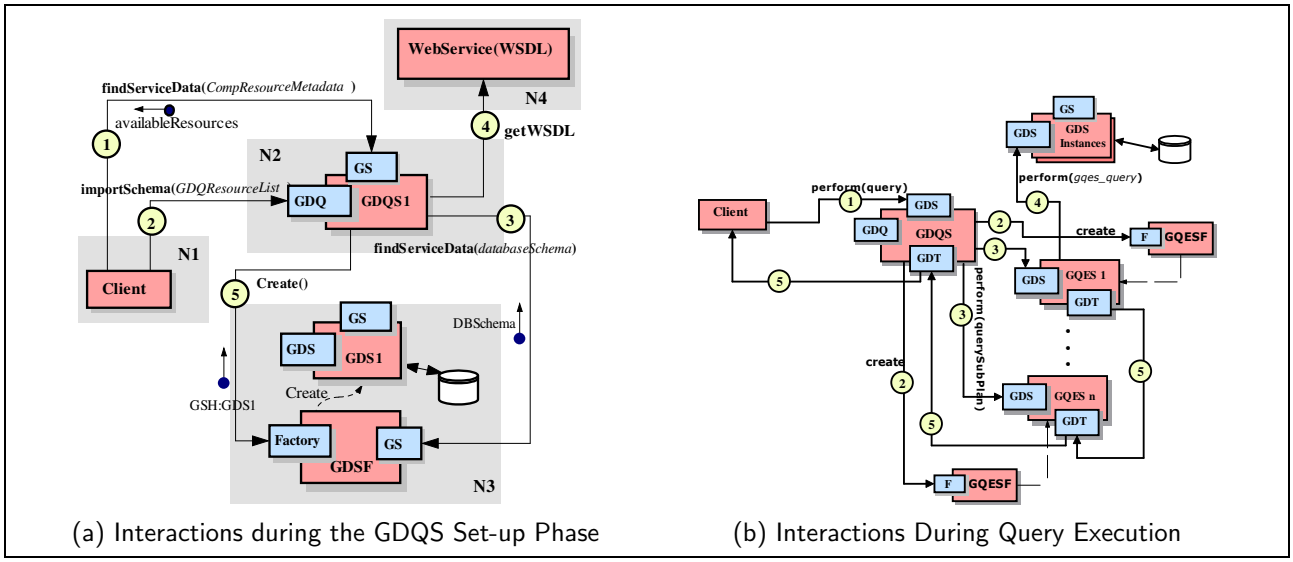
sources in astronomy. This specialisation to a domain and a query type make it difficult to compare SkyQuery with generic middleware like OGSA-DQP or Garlic. The effect is to make OGSA-DQP more robust in general, and SkyQuery distinctively appropriate to the particular user community it targets. Other dimensions along which OGSA-DQP and SkyQuery are difficult to compare stem from differences between Web services and Grid services, e.g., the fact that the latter allow dynamic resource allocation.

ObjectGlobe [3] very much resembles OGSA-DQP in terms of issues addressed and solutions proposed. It aims to provide an open and extensible system that facilitates not only data integration but also dynamic integration of user-defined functions (which are similar to the analysis services that a query might invoke in OGSA-DQP) in a single framework via a declarative query language. The primary difference between ObjectGlobe and OGSA-DQP is that the former aims to be a distributed query processor for the Internet, not the Grid, and this has implications in the way resource virtualisation is achieved. While OGSA-DQP relies on standard service-oriented architectures (e.g., WS/OGSA, OGSA-DAI), ObjectGlobe proposes its own architecture based on Java technologies. In contrast, OGSA-DQP is internally made of and, more importantly, is seen by users as a Grid service. Thus, it can participate in Grid service orchestrations as seamlessly as Grid services that provide much lower-level functionality (e.g., data transport).

5 OGSA-DQP

OGSA-DQP adopts what is, essentially, a mediation/wrapper approach. It can be seen as a mediator component over OGSA-DAI wrappers, except that its main concern is efficient query execution rather than reconciliation of data source heterogeneity. Thus, rather than aiming primarily at projecting to users a global schema, OGSA-DQP query execution plans focus on allowing for evolving circumstances to be taken into account when the optimiser decides on distribution and scheduling. In addition, OGSA-DQP uses a parallel physical algebra while most mediator-based query processors do not.

A distinctive characteristic of OGSA-DQP is its ability to map a query evaluation plan to a lightweight orchestration of implicitly parallel evaluation services that execute over distributed OGSA-DAI resources and web services. Importantly, the exploitation of these resources is achieved via a uniform architecture that enables their virtualisation (i.e., a service-based architecture), rather than through a set of bespoke mechanisms. In addition, as the user provides the query request in a declarative form, the translation of the request into an execution plan is directed by informed decisions of an optimiser component that uses metadata collected from participating virtualised resources. Thus, while from a user point of view OGSA-DQP is seen as a standard database engine with a global schema and a query interface, behind the scenes it dynamically constructs a sophisticated query execution engine to fulfill the request for a query to be evaluated.



(a) Interactions during the GDQS Set-up Phase (b) Interactions During Query Execution

Figure 2: GDQS-GDQES Interactions

OGSA-DQP achieves this functionality by implementing two separate Grid Services that collaborate in the achievement of the overall goal, viz., a Grid Distributed Query Service (GDQS) and a Grid Query Evaluation Service (GQES). A GDQS exposes the primary interaction interfaces for users and acts as a coordinator between

the underlying query compiler/optimiser engine and the GQES instances. A GQES, on the other hand, is used to evaluate (i.e. execute) a query sub-plan assigned to it by the GDQS. To perform a request, a GDQS spawns as many GQESs in as many hosts as stipulated by the partitioning and scheduling policies of the GDQS recommended for that request.

A typical query session with OGSA-DQP starts with a GDQS instance creation and a set-up phase (illustrated in Figure 2.(a)) during which the GDQS instance (1) collects metadata about the capabilities of computational resources, (2)-(3) queries the relevant SDEs of GDS Factories to collect schemas of the data sources, (4) interacts with WSs that front-end analysis capabilities to collect information on supported operations, and (5) instantiates GDSs for the required data sources.

The query session can then continue with multiple query requests to the same GDQS instance. As illustrated in Figure 2.(b)), for each request a GDQS instance (1) uses the compilation, optimisation, partitioning and scheduling capabilities of the Polar* optimiser to generate a distributed query plan optimised for specific requirements of the submitted query, (2) commands the creation of GQESs as stipulated by the partitioning and scheduling decided on by the compiler, and (3)-(5) co-ordinates the GQESs into executing the plan, which effectively constructs a tree-like data flow system with the GDQS instance at the root, GDS instances at the leaf and a collection of GQESs in the middle.

For reasons of space, this paper cannot provide a detailed requirements analysis, but a summary of typical user and data provider requirements in the context of distributed query processing that OGSA-DQP responds to now follows. User requirements include performance and quality of service, ease of use, and extensibility and flexibility. Data provider requirements include data access and control, visibility and ease of provision, and scalability.

In terms of performance and quality of service, users expect reliably low response times and efficiency in (potentially paid-for) resource allocation and use. With regard to usability, it is desirable to have a standard way of discovering and invoking services. In addition, an easy-to-use client API and well-defined interfaces to plug into from front-end systems are also very desirable. Finally, regarding extensibility and flexibility, users benefit from the ability to incorporate new resources dynamically into the data federation, especially when schematic conflict resolution is provided. Another important requirement in this category is the ability to combine data analysis with data retrieval without significant impediments.

In terms of data access and control, data providers should be ensured that access control, user authentication and anonymity are properly supported. With regard to visibility and ease of provision, it is desirable that the framework enables the incorporation of a new resource and ensures that it is visible and accessible by potential users. Finally, regarding scalability, the framework should incorporate mechanisms that allow a provider to cope with increasing system load (e.g., a large number of simultaneous requests, high levels of memory consumption by a single request, etc.)

The design of the OGSA-DQP aims to meet the requirements outlined above primarily by relying on standards-driven OGSA and OGSA-DAI infrastructures as much as possible, while building in added-value in the form of optimised scheduling of distributed data processing tasks and on-demand service composition and orchestration. The main features of the OGSA-DQP in terms of the design principles adopted can be outlined as follows.

Resource virtualization is achieved through a service-oriented architecture, very much in contrast with Garlic and ObjectGlobe, and in line with SkyQuery. Service-oriented architectures provide a convenient model for accessing both computational resources and data resources via a standard interface in a well-defined way. As such, data resource discovery becomes possible using service registries, while computational resource discovery becomes possible via resource indexing services (OGSA-DQP uses GT3 Index Service to access such information). Data resource virtualisation is achieved via GDSs provided by the OGSA-DAI framework. Thus, OGSA-DQP uses GT3 core services for resource virtualisation and management.

Dynamically extensible data processing and service composition layer are used to cope with (and make the most of) evolving computational environment conditions, very much in contrast with Garlic and SkyQuery, and in line with ObjectGlobe. By acquiring and manipulating data in a data-flow architecture that is constructed dynamically, the DQP system constructs on the fly a lightweight distributed query execution engine. It is lightweight in the sense that the resources required for query plan execution are acquired at run time, by instantiating GQESs, and disposed of on a per-query basis.

Internally, the query evaluation functionality is realised by an orchestration of coarse-grained, loosely-coupled services via document oriented interfaces.

While schema integration and conflict resolution are not provided for, the design is such that any data source integration approach (e.g., local-as-view, global-as-view, etc.) can be accommodated if it is, itself, cast as

another service in the growing family of data access and integration services exemplified by OGSA-DAI and OGSA-DQP.

6 Discussion

The OGSA-DQP design aims to provide a generic framework and seeks to exploit effectively the available of Grid resources and the dynamism with which they can be allocated. The ultimate goal is to deliver significant benefits in execution efficiency but in a usable, flexible, intereoperable way. This stands in contrast with pre-configured data federations that delegate most of the data processing to data servers and only control data flows.

Greater benefits are possible if data is replicated or partitioned over multiple locations, because then parallel data access and parallel execution of joins provide maximum value. Likewise, the existence of multiple instances of the same analysis service (or the ability to create instances of an analysis service dynamically) would also boost the performance because it would be possible to stream data simultaneously to multiple services for processing[1]. Thus, the benefits of OGSA-DQP would be more visible where such replicated data sources and services exist and where large amounts of complex data require complex processing.

The OGSA-DQP controls the lifetime of the GDSs and GQESs it creates to ensure that GDS instances are re-used across the queries within the same query session, whereas GQES instances are created and destroyed per query. This model trades off ease of service implementation with number of services to be created (in the case of GQES), but also reduces the cost associated with GDS instance creation in a GDQS session. One could envisage adopting a different model which could be more/less dynamic in terms of GDS or GQES creation.

OGSA-DQP assumes that computational resources can be acquired and used dynamically, which implies that application staging is possible. For true application staging, sophisticated policies are required that guarantee a proper binding of the staged services to service providers, both in legal terms and in technical terms (resource provision, security etc.) Since currently true application staging is not supported by GT3, OGSA-DQP is reduced to assuming that GQES factories already exist on the required Grid nodes. Consequently, it is implied that the service provider which hosts an OGSA-DQP GDQS either has multiple servers with many data sources and evaluators installed on them (which is less likely), or service level agreements with a set of GDS providers as well as computational resource providers to host evaluator factories. This, in turn, implies that business-to-business agreements are already in place.

One performance bottleneck is the lack of an efficient data transfer mechanism. As OGSA-DQP is essentially a data flow system, efficient data movement between the nodes of the computation is critical. Currently, the underlying OGSA-DAI GDS technology is not mature enough to provide this capability. For example, block delivery is not as efficient as it needs to be, because, firstly, block sizes are not configurable, and, secondly, the data is transferred as XML documents over SOAP/HTTP.

7 Conclusions

OGSA-DQP is a distributed query processor exposed to users as an OGSA-compliant Grid service in which query execution plans are themselves service orchestrations. Its use of reference implementations of emerging GGF standards allow it to exploit the advantages of a service-oriented architecture while making the most of dynamic, distributed data and computational resources.

The most distinctive contribution of OGSA-DQP is to demonstrate how Grid services provide a platform for the development of higher-level services aggregating many layers of complex functionality and reaping the benefits of dynamic resource discovery and allocation. In the experience of the development team, its success in doing so (and the advances it makes on previous and current work on distributed query processing) serves as evidence of the latent potential of the Grid services vision, in spite of shortcoming that are all but inevitable in such a nascent framework.

Acknowledgements The work reported in this paper has been supported by grants from the UK DTI and the EPSRC through its Core e-Science Programme in the OGSA-DAI and the myGrid projects. Their support

[1] At present, other than the GDSs, the services invoked from OGSA-DQP have been web services, as these are currently more widely deployed; the OGSA-DQP infrastructure is essentially in place that would enable dynamic creation of Grid analysis services for parallel invocation.

is gratefully acknowledged.

References

[1] M.Nedim Alpdemir, Arijit Mukherjee, Norman W. Paton, Paul Watson, Alvaro A.A. Fernandes, Anastasios Gounaris, and Jim Smith. Ogsa-dqp: A service-based distributed query processor for the grid. In Simon J. Cox, editor, *Proceedings of UK e-Science All Hands Meeting Nottingham*. EPSRC, 2–4 September 2003.

[2] M. Antonioletti, N.P. Chue Hong M.Atkinson, A.Krause, S. Malaika, G. McCance, S. Laws, J Magowan, N.W. Paton, and G.Riccardi. Grid Data Service Specification. Technical report, DAIS-WG, Global Grid Forum, 2003. Draft, June 6, 2003.

[3] Reinhard Braumandl, Markus Keidl, Alfons Kemper, Donald Kossmann, Stefan Seltzsam, and Konrad Stocker. Objectglobe: Open distributed query processing services on the internet. *Bulletin of the Technical Committee on Data Engineering*, 24(1):65–70, March 2001.

[4] A. Anjomshoaa et al. The design and implementation of grid database services in ogsa-dai. In Simon J. Cox, editor, *Proceedings of UK e-Science All Hands Meeting Nottingham*. EPSRC, 2–4 September 2003.

[5] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, OGSI-WG, Global Grid Forum, 2002. Draft 2.9, June 22, 2002.

[6] V. Josifovski, P. Schwarz, L. Haas, and E. Lin. Garlic: A New Flavor of Federated Query Processing for DB2. In *Proc. SIGMOD*, pages 524–532, 2002.

[7] Tanu Malik, Alex S. Szalay, Tamas Budavari, and Ani R. Thakar. SkyQuery: A Web Service Approach to Federate Databases. In *Proc. CIDR*, 2003.

[8] Mary Tork Roth and Peter M. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 266–275. Morgan Kaufmann, 1997.

[9] Thomas Sandholm and Jarek Gawor. Globus Toolkit 3 Core A Grid Service Container Framework. Technical report, 2003. www-unix.globus.org/toolkit/3.0/ogsa/docs/.

[10] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 22(3):183–236, 1990.

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

[12] J. Smith, S. F. M. Sampaio, P. Watson, and N. W. Paton. Polar: An architecture for a parallel odmg compliant object database. In *Conference on Information and Knowledge Management (CIKM)*, pages 352–359. ACM Press, 2000.

[13] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T.Sandholm, D.Snelling, and P.Vanderbilt. Open Grid Service Infrastructure (OGSI). Technical report, OGSI-WG, Global Grid Forum, 2003. Version 1.0.