Databases and the Grid: JDBC in WSDL, or something altogether different?

Norman W. Paton

Department of Computer Science, University of Manchester,
Manchester, M13 9PL, UK
norm@cs.man.ac.uk

Abstract. The Grid is rising to prominence both as a vision and as an infrastructure. The vision is of dynamically formed virtual organisations that collaborate to address shared goals. The infrastructure, although still a work in progress, is a middleware that facilitates the sharing of networked resources on a global scale. Databases are important resources that are central to many organisations, and as such must be able to be accessed, integrated and managed in a Grid environment. How should this be done? What problems must be overcome when working in a Grid environment? What new expectations, opportunities and patterns of use come to the fore? This presentation both discusses core grid data access services and the wider implications and opportunities of Grids for database technologies.

1 Introduction

The Grid [10] is rising to prominence both as a vision and as an infrastructure. The vision is of dynamically formed virtual organisations that collaborate to address shared goals. Although often associated with scientific applications, for example in particle physics and astronomy, the need for incremental, secure, easily managed and efficient resource sharing across organisations is present whenever collaboration takes place, so the potential reach of the Grid is extremely broad. The infrastructure, although still a work in progress, is a middleware that facilitates the sharing of networked resources on a global scale. There is no single Grid middleware; there are several commercial (e.g., Avaki (`www.avaki.com`), Platform (`www.platform.com`)) and not for profit (e.g., Globus (`www.globus.org`), UNICORE (`www.unicore.org`)) offerings, although Globus [12] is widely credited with providing the principal proving ground for many of the core functionalities that Grids require.

The name, and much of the original focus, was based on the idea of a global computational Grid, analogous to an electricity Grid, which would provide users with transparent access to remote computing resources. A CPU-hungry application could be moved to a suitable, available computer and executed without the need for micromanagement by the user. As the set of Grid-enabled resources, and their loads, will change over time, so decisions on resource selection must be made dynamically. This raises the possibility of computational marketplaces, in

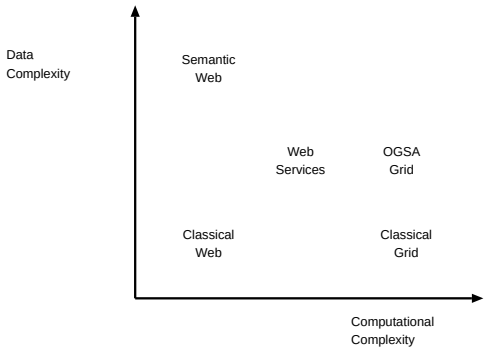

Fig. 1. The grid in context.

which service providers offer computational resources, and users select a resource based on criteria including price and quality of service offered.

The initial application drivers for the development of a Grid infrastructure were in large-scale scientific and engineering applications. For example, the first production grid was probably NASA's Information Power Grid (IPG) [19], which was developed in the context of a vision to simulate the US national air space, including, for example, detailed models of airframes and landing gear. Supporting such an application involves the integration of many existing computational resources, and the provision of a large collection of inter-related services to help in the management of the resources. These services include security services, uniform data access services, global event services, co-scheduling, etc. There has also been considerable interest in the use of Grids in Particle Physics (e.g., [18]) and Astronomy (e.g., [26]).

As these initial Grid applications were often closely associated with devices or tools that read and/or generated flat files, support for files rather than for the management of structured data had the highest profile in early Grid toolkits. However, file management systems and the registries associated with Grid toolkits themselves quickly became complex, and database management systems were increasingly used to store metadata (e.g., [7]). However, where databases are currently used as components within Grid toolkits, it is rarely the case that the database is a first-class citizen – for example, the Grid authentication and data transfer mechanisms may not be closely integrated with those of the database. As such, early Grid toolkits can be seen as providing effective facilities for accessing and sharing low-level data and computational resources, but to provide only limited support for complex data, as illustrated in Figure 1. The figure illustrates the level of support provided by different internet scale software architectures for managing the data and computational complexity of applications. In essence, early Grid middlewares provided operating system style functionalities, in a distributed and heterogeneous environment, over which higher-level application-oriented functionalities could be built.

It became clear, however, that the capabilities being developed for use in Grid middlewares overlapped with those that are under development in the web services community to support e-Business transactions (e.g., for authentication, protocol-neutral communication, resource description, etc). As such, a proposal has been made for a service-oriented architecture for the Grid, known as the Open Grid Services Architecture (OGSA) [11]. While the specific services that will constitute such an architecture are not yet fully understood, the OGSA will include interfaces for capabilities such as resource description, monitoring, negotiation, deployment, fault management and logging. As such, the OGSA will make available classical grid functionalities using the service description and invocation standards of the web services community, as illustrated in Figure 1. The OGSA architecture will form the core of activity within the Global Grid Forum (GGF) (`www.ggf.org`), the principal standards body for Grid computing. As databases are important resources that need to be deployed, accessed and managed within a Grid setting, the Database Access and Integration Services Working Group (DAIS-WG) has been set up within the GGF to develop standards for service-based interfaces to databases. It is not yet clear precisely what will occupy the top right hand corner of Figure 1, but work is underway to identify how semantic web technologies can be applied to the description of Grid resources, to yield a "semantic grid" [15].

This paper discusses the relationship between databases and the Grid under two headings:

1. *Deploying databases on the Grid:* identifying how databases can be made available as Grid resources.
2. *Deploying database technologies on the Grid:* identifying how functionalities and abstractions developed in the database community can be deployed in a Grid setting to provide higher level data access and integration services.

As such, with reference to the title, this paper discusses both core grid data access services ("JDBC in WSDL", to a first approximation) and the wider implications and opportunities of Grids for database technologies ("something altogether different"). Henceforth, the focus will largely be on service-based Grids, as the momentum behind the integration of web and Grid services seems great. The remainder of the paper is structured as follows. Section 2 discusses the requirements for database access in a Grid setting, outlines several proposals that have been made, and provides some concrete examples form the OGSA-DAI project. Section 3 discusses the role that Grids can play in supporting the development of higher level data integration and provisioning, and provides some concrete examples from the OGSA-DQP project. Some conclusions and pointers to future work are presented in Section 4.

2 Deploying Databases on the Grid

Deploying databases on the Grid means that databases are supported as first class citizens in a Grid setting. What does it mean for databases to be "first

class citizens"? As in life, citizenship brings with it both rights and responsibilities. In a Grid setting, a database access middleware might expect to be able to benefit from other Grid functionalities, such as for service discovery and data movement, but might also be expected to conform to community norms in areas such as authentication, monitoring and transaction management. As such, a database access service could potentially implement a significant number of web and grid service interfaces, few of which have been formally standardised at the time of writing. Relevant proposals in the web services arena include those for transaction management (e.g. WS-Transaction [6] or WS-TXM [4]) and for context management (e.g. WS-Coordination [5] or WS-Context [4]), and in the Grid Services arena include many of the capabilities identified in the OGSA Architecture proposal (`https://forge.gridforum.org/projects/ogsa-wg`). However, as such specifications are intended to be composable, it is possible to discuss the specific operations and access patterns of a data access service with minimal direct reference to the potentially numerous related specifications, on the assumption that a database service will also implement, for example, a transaction management interface, and thereby behave in the manner required by the corresponding specification.

The behaviour of a data access service is characterised in [13] as having the following principal aspects:

1. *Data Description:* the description of the key characteristics of the data made available by the service.
2. *Data Access:* the interface that permits access to and modification of the data made available by the service.
3. *Data Factory:* the interface by which a new data resource can be derived from an existing resource.
4. *Data Management:* the interface by which operations manage the relationship between the data access service and the underlying data management infrastructure.

Data Description is important in service-oriented architectures as, at least in principle, it should be possible to dynamically discover, interrogate and make use of a service from a running application. For example, in the case of databases, a data federation could be dynamically constructed from the data services in an organisation that record information on customers. As such, services should be able to make available enough information about themselves to allow an informed choice to be made as to their suitability for a specific purpose. Use cases for data description in a Grid setting are described in [22].

Data Access is important, as interfaces must be developed that allow remote requesters to make convenient and efficient use of a data resource. Traditionally, database access interfaces assume a client-server interaction model. However, it is possible to envisage a wide range of different access patterns that are equally valid in a Grid setting. For example, the usage scenarios illustrated in Figures 2 and 3 are not only valid, but quite common in distributed applications [25]. In Figure 2, a query evaluation request is sent to a data service, but the result is delivered to a party other than the requester. The third party recipient of

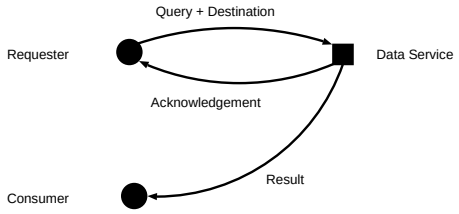

Fig. 2. A data access request with third party push delivery.

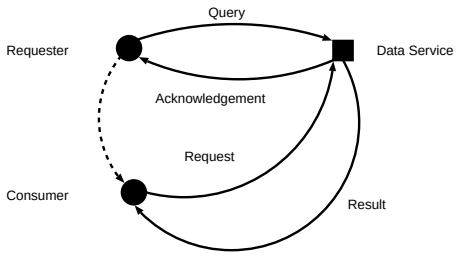

Fig. 3. A data access request with third party pull delivery.

the result could, for example, be a service through the use of a standard data delivery interface, or could be a file in a remote filesystem. In Figure 3, the result is not so much delivered to, but obtained by, a third party consumer. In this scenario, a way of identifying the result is passed from the requester to the consumer, who then invokes operations on the data service to access the result. As such, there might be somewhat more to data access in a Grid setting than at first meets the eye – although JDBC provides a useful guide to the kinds of capability that are at the core of data access, as a client library for client-server access it provides less comprehensive functionality than one might hope to see in grid data access services.

Data Factory functionality is important, as it is often necessary for services to create data resources that are to some extent independent of the interaction or the resources that brought them into being. As Grid middleware is very much focused on computational and data resources, there has been considerable discussion as to how best to represent networked resources in a service-oriented architecture. The most recent proposal in this regard is the Web Services Resource Framework [9], which characterises a WS-Resource as an association between a web service and some state that is accessed through the web service. In a data services context, a WS-Resource could be used, for example, to represent the result of a query evaluated by a requester who wants to make the result known to a third party consumer, as in Figure 3.

Data Management is important because the coordinated deployment and maintenance of a collection of data services in a distributed environment requires systematic support for management of the services. It is anticipated that groups

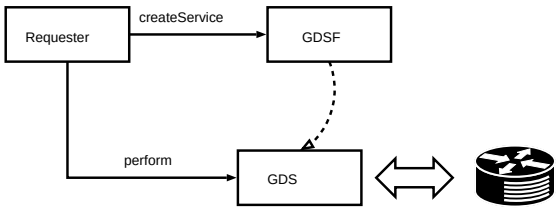

Fig. 4. Basic service interactions in OGSA-DAI.

such as the Web Services Distributed Management Working Group of OASIS (`http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm`) will provide generic interfaces both for managing services and for managing resources using services.

As such, the development of data access services in the Grid can be seen to touch on a range of design issues, which complement the functionality of initial vendor offerings for web service access to databases (e.g., [21, 23]), which principally addressed a narrower range of functionalities, such as making stored procedures available as web services.

The OGSA-DAI project (`http://www.ogsa-dai.org.uk/`) provides an early example of service-based access to databases [2], and now forms part of the Globus toolkit. Figure 4 illustrates the basic interactions between services in OGSA-DAI using the grid service creation facilities of Globus Toolkit 3. A client (referred to here using web services terminology as a *Requester*), first invokes the *createService* operation on a Grid Database Service Factory (*GDSF*), which leads to the creation of a Grid Database Service (*GDS*). The *GDS* represents a session over a particular database, to which the *Requester* can then submit requests using the *perform* operation.

A single request to a GDS, conveyed as a parameter of *perform*, is capable of evaluating a collection of linked activities. Each activity describes an atomic capability provided by the service. For example, there are activities that can query a database, apply a transformation to a result, or deliver a result to a third party. The request in Figure 5 makes use of *sqlQueryStatement* and *deliverToGFTP* activities to retrieve data from a *frogGenome* database and deliver the result using GridFTP (`http://www.globus.org/datagrid/gridftp.html`) to a third party, thereby implementing the scenario illustrated in Figure 2. The result of the `sqlQueryStatement`, which is named *statementresponse*, is explicitly used as input to the `deliverToGFTP` activity. As such, OGSA-DAI provides a coarser-grained interface than JDBC, providing a means of grouping data movement, database access and data transformation operations in a single request.

The deployment of databases in service-based grids can thus be seen to raise a wide range of design issues, relating, for example, to the granularity of requests, the support provided for handling of query results, the delivery scenarios supported directly by the data access service, and the level of detail provided for service description. Furthermore, as Grids are often concerned with the timely

```
<gridDataServicePerform ...>
  <request name="requestAsynchPush">

    <sqlQueryStatement name="statement">
      <dataResource>frogGenome</dataResource>
      <expression>
        select * from chromFrag where num = 30
      </expression>
      <webRowSetStream name="statementresponse"/>
    </sqlQueryStatement>

    <deliverToGFTP name="d1">
      <fromLocal from="statementresponse"/>
      <toGFTP host="ogsdai.org.uk"
              port="8080" file="path/to/myfile.txt"/>
    </deliverToGFTP>

  </request>
</gridDataServicePerform>
```

Fig. 5. An OGSA-DAI perform document for a data access request with third party push delivery.

provisioning of data, it can also be speculated that subscription-based data access [8] is likely to have a more prominent role in Grid computing than it has to date. The Information Dissemination Working Group of the GGF is developing proposals in this space.

3 Deploying Database Technologies

Database technologies, techniques and principles can be applied in a Grid setting in various ways. This section makes a case that Grid and database technologies each have something to offer the other, by discussing how the development of data management systems can benefit from Grids in Section 3.1, and how Grids stand to benefit from effective deployment of database technologies in Section 3.2. OGSA-DQP [1] is used as a concrete example of a database middleware developed over a service-based Grid in Section 3.3.

3.1 Database Technologies need the Grid

Database technologies might be considered likely to benefit from Grid technologies for several reasons. For example, database management systems, as complex pieces of software operating in a networked environment, stand to benefit from systematic access to computational and storage resources, and from facilities for dynamic resource discovery and allocation. Furthermore, the study of systems

developed without access to service-based Grids can be used to make the case that Grid services provide capabilities that are required to enable the effective development of information integration platforms.

As an example, ObjectGlobe [3] provided a platform for executing distributed queries, using web technologies to provide access to data stores and computational resources ("cycle providers"). ObjectGlobe consists of a collection of distributed Java components that communicate with each other using messages encoded in XML. In so doing, ObjectGlobe, which largely predated the web and grid service activities, developed registries, database wrappers and cycle providers with a view to supporting distributed query processing, in which query plan partitions could be allocated to widely distributed nodes on the internet. Many of the components that were developed from scratch for ObjectGlobe now form part of a service-based Grid middleware, such as Globus. For example, Globus provide registries for describing both services and the capabilities of computational resources, and is shipped with the OGSA-DAI database wrappers. In addition, it inherits from the web services community consistent techniques for describing and invoking operations on heterogeneous platforms. As such, the developers of distributed query processors for the Grid have significantly less work to do than was required to construct ObjectGlobe over the internet only a few years ago.

The highest profile association to date of Grid technologies with a commercial database management system is in Oracle $10g$ (the "g" is for "grid"). Here the emphasis is on using Grid functionalities to ease the deployment, management and adaptation of the database management system as a complex software system in a distributed environment [17].

As such, data management systems can be seen as complex software artifacts in a distributed, heterogeneous setting, and thus able to benefit from the dynamic, flexible and consistent access to computational resources that the Grid is promising to provide.

3.2 The Grid needs Database Technologies

Database management systems provide high-level facilities for managing large and potentially complex data resources. In so doing, many details as to how and where the data is physically stored are hidden from users. In a distributed setting, database systems further exploit the notion of transparencies to characterise aspects of the management of data that are hidden from users (e.g., the presence of replicas, the language used to access specific portions of the data, etc). This notion of transparencies can also be applied in the design of data grids [24], and the level of abstraction at which Grid resources are accessed can be seen as steadily increasing with time.

The classical representation of a collection of dependent tasks that are to be executed together in a Grid setting is as a Directed Acyclic Graph (DAG), and systems such as Condor can be used to queue and schedule jobs on a Grid [14]. However, such representations tend to refer to jobs and resources in terms of the programs to be executed and the files to be acted on, rather than in terms

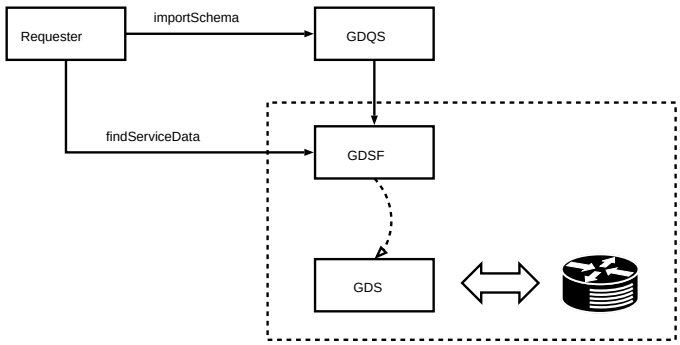

Fig. 6. Initialising a GDQS in OGSA-DQP.

of higher level abstractions. As such, work is underway to develop higher level representations for data intensive tasks. For example, Chimera [11] supports the notion of *virtual data* by describing how a data product is produced, and scheduling the jobs required to produce the data product on demand. With a view to integrating database access and management more fully with this process, the GridDB project [20] has developed a language that combines facilities for database access with those for the invocation of programs on a Grid. In essence, a relational representation of data is used to model data of relevance to an application, and analyses are wrapped to act on tuples that may be initial inputs or intermediate values in a workflow, all of which can be managed using standard relational database facilities. The work described in the next section, on OGSA-DQP, has a similar motivation to that of GridDB, but is acting in the context of service-based Grids. As such, a key difference is that OGSA-DQP uses existing service interfaces to access remote resources, and thus does not require custom wrapping of analysis programs.

3.3 OGSA-DQP as a Case Study

OGSA-DQP (`http://www.ogsa-dai.org.uk/dqp/`) is a service-based distributed query processor for the Grid, which supports the evaluation of queries over one or more OGSA-DAI wrapped databases and other web services [1]. In essence, OGSA-DQP allows a single query to access data from multiple resources, and to invoke web service operations on data extracted from database services. As such OGSA-DQP can be seen as providing declarative service orchestration, in a manner that complements that of, for example, workflow systems.

Figure 6 illustrates the initialisation of a Grid Distributed Query Service (*GDQS*). It is assumed that the *GDQS* service has already been created in a similar manner to that illustrated for an OGSA-DAI GDS in Figure 4. The *Requester* then identifies a number of Grid Database Service Factories (*GDSF*s), for example by looking them up in a registry, that are associated with databases that may be required to participate in a distributed query processing activity.

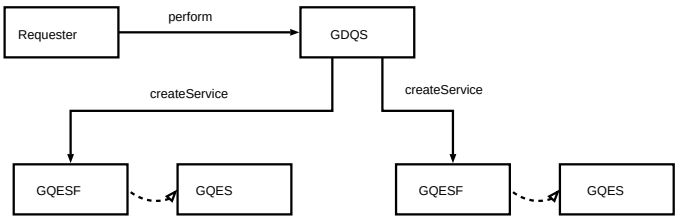

Fig. 7. Creating evaluators for a specific query in OGSA-DQP.

The specific properties of a database that can accessed using a *GDS* created by a *GDSF* can be ascertained by invoking the GDSF's *findServiceData* operation. When a suitable *GDSF* has been identified, the *Requester* can then invoke the *importSchema* operation on the *GDQS*, which leads to a *GDS* being created that is used to access the relevant database by queries submitted to the *GDQS*.

Once the resources have been identified over which queries are to be run, specific queries can be submitted by the *Requester* to the *GDQS* using the *perform* operation, as illustrated in Figure 7. The *GDQS*, as described in [1], compiles the query, to yield a collection of query partitions, each of which must be evaluated using a Grid Query Evaluation Service (*GQES*). Thus the *GDQS* then creates a collection of *GQES*s, as illustrated in Figure 7, to each of which is assigned a single partition. Invoking *perform* on the root evaluator leads to a cascaded collection of evaluation requests, which execute the query in a distributed fashion, potentially with both pipelined and partitioned parallelism.

The deployment of database technologies in a Grid setting can thus be seen as having implications both for the Grid and for the way in which the technology is used. In OGSA-DQP, data intensive tasks that involve access to multiple data and computational resources can be expressed using queries, providing a new, more declarative mechanism for describing data intensive requests over the Grid. In addition, parallel database query processing technology is adapted to work in a highly heterogeneous setting, where the Grid middleware provides useful abstractions for hiding details of the resources used in the evaluation of requests. Thus database technologies can have a significant role as part of the Grid middleware stack, and Grids can impact on the way database technologies are engineered, deployed and used.

4 Conclusions

So, are databases for the Grid principally JDBC in WSDL, or something altogether different? Well, there is a need for service-based interfaces to databases for use as part of the Grid middleware stack, which in many cases will involve functionality that is familiar from existing database access middlewares such as JDBC. However, as discussed in Section 2, it is possible to identify scenarios and thus opportunities to explore somewhat more expressive data access interfaces, both for request-response and publish-subscribe access to data. In addition, as

discussed in Section 3, it is possible to envisage a much closer association of Grid and data management technologies. This can be not only to increase the flexibility and manageability of existing database software, but also to allow database infrastructure to be deployed more widely for information and process integration in Grids. Although initial work has taken place to deploy database technologies for service orchestration in Grids, as in GridDB and OGSA-DQP, this work can be seen as being at an early stage, with scope for significant additional developments. For example, wide deployment of queries for service orchestration in Grids is likely to require further work on adaptive query processing to take account of the unpredictability of resource availability and computational costs associated with Grids [16], and providing automatic support for varying quality of service requirements is likely to involve significant additional work on how these requirements are both described and supported.

Acknowledgements: The work reported in this paper has emerged from ongoing interactions in the OGSA-DAI, myGrid and OGSA-DQP projects, and in the DAIS Working Group of the GGF. Research on grid data management is supported at Manchester by the Engineering and Physical Sciences Research Council and the UK e-Science Programme.

References

1. N. Alpdemir, A. Mukherjee, N. W. Paton, P. Watson, A. A. A. Fernandes, A. Gounaris, and J. Smith. Service-based distributed querying on the grid. In *Proc. of ICSOC*, pages 467–482. Springer-Verlag, 2003.
2. A. Anjomshoaa et al. The design and implementation of grid database services in ogsa-dai. In S. Cox, editor, *Proc. UK e-Science Programme All Hands Meeting*, pages 795–801. http://www.nesc.ac.uk/events/ahm2003/AHMCD/ahm_proceedings_2003.pdf, 2003.
3. R. Bramandl, M. Keidl, D. Kossman, A. Kreutz, S. Seltzsam, and K. Stocker. ObjectGlobe: Uniquitous query processng on the Internet. *VLDB Journal*, 10:48–71, 2001.
4. D. Bunting et al. Web Services Composite Application Framework (WS-CAF) Version 1.0. Technical report, Arjuna Technologies Ltd., Fujitsu Limited, IONA Technologies Ltd., Oracle Corporation and Sun Microsystems Inc., http://developers.sun.com/techtopics/webservices/wscaf/primer.pdf, 2003.
5. L. F. Cabrera et al. Web Services Coordination (WS-Coordination). Technical report, IBM developerWorks Report, ftp://www6.software.ibm.com/software/developer/library/ws-coordination.pdf, 2002.
6. L. F. Cabrera et al. Web Services Transaction (WS-Transaction). Technical report, IBM developerWorks Report, http://www-106.ibm.com/developerworks/library/ws-transpec/, 2002.
7. P. Dinda and B. Plale. A unified relational approach to grid information services. Technical Report GWD-GIS-012-1, Global Grid Forum, 2001.
8. P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(3):114–131, 2003.

9. I. Foster et al. Modeling Stateful Resources with Web Services. Technical Report Version 1.1, Computer Associates International, Fujitsu Limited, Hewlett-Packarg Development Company, IBM, University of Chicago, http://www.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf, 2004.
10. I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing*. Morgan Kaufmann Publishers Inc., 2004.
11. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid Services for Distributed System Integration. *IEEE Computer*, 35:37–46, 2002.
12. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. Supercomputer Applications*, 15(3), 2001.
13. I. Foster, S. Tuecke, and J. Unger. OGSA Data Services. Technical Report Informational Draft, Global Grid Forum, Database Access and Integration Services Working Group, https://forge.gridforum.org/projects/dais-wg/document/OGSA_Data_Services-ggf9/en/1, 2003.
14. J. Frey, T. Tannembaum, I. Foster, M. Livney, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. In *Proc. 10th Int. Conf. on High Performance Distributed Computing*, pages 55–66. IEEE Press, 2001.
15. C. Goble, D. DeRoure, N. Shadbolt, and A. Fernandes. Enhancing Services and Applications with Knowledge and Semantics. In I. Foster and K. Kesselman, editors, *The Grid 2: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 2004.
16. A. Gounaris, N. Paton, A. Fernandes, and R. Sakellariou. Adaptive query processing and the grid: Opportunities and challenges. In *Proc. 1st Intl. Workshop on Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems*. IEEE Press, 2004. To be published.
17. B. Goyal. Oracle Database 10*g* - The Database for the Grid. Technical Report 40123, Oracle White Paper, http://otn.oracle.com/tech/grid/collateral/10gdbgrid.pdf, 2003.
18. G. Graham, R. Cavanaugh, P. Gouvares, A. D. Smet, and M. Livney. Distributed Data Analysis: Federated Computing for High-Energy Physics. In I. Foster and K. Kesselman, editors, *The Grid 2: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 2004.
19. W. Johnston, D. Gannon, and B. Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. In *8th IEEE Symposium on High Performance Distributed Computing*. IEEE Computer Society, 1999.
20. D. Liu and M. Franklin. GridDB: A Data-Centric Overlay for Scientific Grids. Technical Report CSD-04-1311, UC Berkeley, http://www.cs.berkeley.edu/ dtliu/pubs/griddb_tr.pdf, 2004.
21. S. Malaika, C. Nelin, R. Qu, B. Reinwald, and D. Wolfson. DB2 and Web Services. *IBM Systems Journal*, 41(4):666–685, 2002.
22. S. Malaika and N. Paton. CIM Database Model for Data Access and Integration Services: Scenarios. Technical Report Informational Draft, Global Grid Forum, Database Access and Integration Services Working Group, https://forge.gridforum.org/projects/dais-wg/document/DAIS_CGS_Scenarios-ggf10/en/1, 2004.
23. K. Mensah and E. Rohwedder. Database Web Services. Technical Report Informational Draft, Oracle Corporation White Paper, http://otn.oracle.com/tech/webservices/htdocs/dbwebservices/Database_Web_Services.pdf, 2002.

24. V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, and C. Baru. Services for Data Access and Data Processing on Grids. Technical Report GFD.14, Global Grid Forum, http://www.ggf.org/documents/GWD-I-E/GFD-I.014.pdf, 2003.
25. G. Riccardi, M. Subramanian, S. Misra, and S. Laws. DAIS Usage Scenarios. Technical Report Informational Draft, Global Grid Forum, Database Access and Integration Services Working Group, https://forge.gridforum.org/projects/dais-wg/document/DAIS_Usage_Scenarios-ggf10/en/1, 2004.
26. A. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, and D. R. Slut. Designing and mining multi-terabyte astronomy archives: The sloan digital sky survey. In *Proc. ACM SIGMOD*, pages 451–462. ACM Press, 2000.