

# Grid Database Access and Integration: Requirements and Functionalities

Malcolm P Atkinson, UK National e-Science Centre. mpa@dcs.gla.ac.uk.

Vijay Dialani, University of Southampton. vkd00r@ecs.soton.ac.uk.

Leanne Guy, CERN. Leanne.Guy@cern.ch.

Inderpal Narang, IBM Almaden Research Center. narang@almaden.ibm.com.

Norman W Paton, University of Manchester. norm@cs.man.ac.uk<sup>1</sup>.

Dave Pearson, Oracle UK. dave.pearson@oracle.com.

Tony Storey, IBM Laboratories, Hursley. tony\_storey@uk.ibm.com.

Paul Watson, University of Newcastle upon Tyne. Paul.Watson@ncl.ac.uk.

Draft of July 4, 2002

## 1 Introduction

This document seeks to provide a context for the development of standards for Grid Database Access and Integration Services (DAIS), with a view to motivating, scoping and explaining standardization activities within the DAIS Working Group of the Global Grid Forum (GGF) (<http://www.cs.man.ac.uk/grid-db>). It is hoped that the DAIS Working Group will lead to the creation of one or more standard specifications for services that can be used to ease the deployment of data-intensive applications within the Grid, and in particular applications that require access to database management systems (DBMSs) and other stores of structured data. To be effective, such standards must:

1. Address recognized requirements.
2. Complement other standards within the GGF and beyond.
3. Have broad community support.

The hope is that this document can help with these points by: (1) making explicit how requirements identified in Grid projects give rise to the need for specific functionalities addressed by standardization activities within the Working Group; (2) relating the required functionalities to existing and emerging standards; and (3) involving widespread community involvement in the evolution of this document, which in turn should help to inform the development of specific standards. In terms of (3), it is intended that this document be revised in the light of discussions in the run-up to GGF6. It is anticipated that the list of authors will grow over this period, as the intention is to have direct contributions from a wide range of interested parties.

This document deliberately does not propose standards – its role is to help in the identification of areas in which standards are required, and for which the GGF (and in particular the DAIS Working Group) might provide an appropriate standardisation forum.

The remainder of the document is structured as follows. Section 2 introduces various features of database access and integration services by way of a scenario. Section 3 introduces the requirements

---

<sup>1</sup> Corresponding author.

for Grid database services. Section 4 summarizes key functionalities associated with database access and integration. Section 5 presents some conclusions and pointers to future activities. A current weakness of the document is that the requirements from Section 3 are not explicitly related to the functionalities in Section 4. This will be addressed in future revisions of the document.

## 2 Overview of Database Access and Integration Services

This section uses a straightforward scenario to introduce various issues of relevance to database access and integration services. A service requestor needs to obtain information on *proteins* with a known *function* in *yeast*. The requestor may not know what databases are able to provide the required information. Indeed, there may be no single database that can provide the required information, and thus accesses may need to be made to more than one database. The following steps may need to be taken:

1. The requestor accesses an *information service*, to find database services that can provide the required data. Such an enquiry involves access to *contextual metadata* [Pearson 02], which associates a concept description with a database service. The relationship between contextual metadata and a database service should be able to be described in a way that is independent of the specific properties (e.g., the data model) of the database service.
2. Having identified one or more database services that are said to contain the relevant information, the requestor must select a service based on some criteria. This could involve interrogating an information service or the database service itself, to establish things like: (i) whether or not the requestor is authorized to use the service; (ii) whether or not the requestor has access permissions on the relevant data; (iii) how much relevant data is available at the service; (iv) the kinds of information that are available on proteins from the service; (v) the way in which the relevant data is stored and queried at the service. Such enquiries involve *technical metadata* [Pearson 02]. Some such metadata can be described in a way that is independent of the kind of database being used to support the service (e.g., information on authorization), whereas some depends on properties of the underlying database (e.g., the way the data is stored and accessed). Provenance and data quality are other criteria that could be used in service selection, and which could usefully be captured as properties of the source.
3. Having chosen a database service, the requestor must formulate a request for the relevant data using a language understood by the service, and dispatch the request. The range of request types (e.g., query, update, begin-transaction) that can be made of a database service should be independent of the kind of database being used, but specific services are sure to support different access languages and language capabilities [Paton 02]. The requestor should have some control over the structure and format of results, and over the way in which results to a request are delivered. For example, results should perhaps be sent to more than one location or they should perhaps be encrypted before transmission. The range of data transport options that can be provided is largely independent of the kind of database that underpins the service.

The above scenario is very straightforward, and the requestor could have requirements that extend the interaction with the database services. For example, there may be several copies of a database, or parts of a database may be replicated locally (e.g., all the data on yeast may be stored locally by an organization interested in fungi). In this case, either the requestor or the database access service may consider the access times to replicas in deciding which resource to use. It is also common in bioinformatics for a single request to have to access multiple resources, which may in turn be eased by a data integration service [Smith 02]. In addition, the requestor may require that the accesses to different services run within a transactional model, for example, to

ensure that the results of a request for information are written in their entirety or not at all to a collection of distributed database services.

The above scenario illustrates that there are many aspects to database access and integration in a distributed setting. In particular, various issues of relevance to databases services (e.g., authorization and replication) are important to services that are not making use of databases. As such, it is important that the DAIS Working Group is careful to define its scope and evolve its activities taking full account of (i) the wide range of different requirements and potential functionalities of Grid Database Services, and (ii) the relationship between database and other services supported within The Grid.

### **3 Requirements for Grid Database Services**

Generic requirements for data access and integration were identified through an analysis exercise conducted over a three-month period, and reported fully in [Pearson 02]. The exercise used interviewing and questionnaire techniques to gather requirements. Interviews were held and questionnaire responses were received from UK Grid and related e-Science projects. Additional input has been provided by CERN, and by the European Astrowise and DataGrid projects.

#### **3.1 Data Sources and Resources**

The analysis exercise identified the need for the Grid to provide access to data directly from data sources and data resources.

Data sources stream data in real or pseudo-real time from instruments and devices, or from applications that perform *in silico* experiments or simulations. Examples of instruments that stream data include astronomical telescopes, detectors in a particle collider, remote sensors, and video cameras. Data sources may stream data for a long period of time but it is not necessarily the case that any or all of the output streamed by a data source will be captured and stored in a persistent state.

Data resources are persistent data stores held either in file structures or in database management systems (DBMSs). They can reside on-line in mass storage devices and off-line on magnetic media. Invariably, the contents of a database are linked in some way, usually because the data content is common to a subject matter or to a research programme. Throughout this document the term database is applied to any organised collection of data that may span one or more data resources.

The ability to group a logical set of data resources stored at one site, or across multiple sites is an important requirement, particularly for curated data repositories. It must be possible to reference the logical set as a 'virtual database', and to perform set operations on it, e.g. distributed data management and access operations.

#### **3.2 Data Structure and Representation**

In order to support the requirements of all science disciplines, the Grid must support access to all types of data defined in every format and representation. It must also be possible to access some numeric data at the highest level of precision and accuracy; text data in any format, structure, language, and coding system; and multimedia data in any standard or user defined binary format.

#### **3.3 Data Organisation**

Traditionally, data in many scientific disciplines have been organized in application-specific file structures, and structured to optimise compute intensive data processing and analysis. A great deal of data accessed within current Grid environments still exists in this form. However, there is an important requirement for the Grid to provide access to data held in DBMSs and XML repositories.

These technologies are increasingly being used in bioinformatics, chemistry, environmental sciences and earth sciences for a number of reasons. First, they provide the ability to store and maintain data in application independent structures. Second, they are capable of representing data in complex structures, and of reflecting naturally occurring and user defined associations. Third, relational and object DBMSs also provide a number of facilities for automating the management of data and its referential integrity.

### **3.4 Data States**

No attempt was made in the analysis exercise to distinguish between data, information, and knowledge when identifying requirements. This is because one worker's knowledge can be another worker's information or data. However, the analysis exercise did distinguish between each stage in the data life cycle to reflect how data access and data operations vary.

Raw data are created by a data source, normally in a structure and format determined by the output instrument and device. A raw data set is normally accessed sequentially, and during its life may be repeatedly reprocessed. Raw data sets are commonly archived once processing is complete. Therefore, the Grid needs to provide the ability to secure raw data off-line and to restore the data on line.

Reference data are values that rarely change. They are frequently referenced in processing, transforming, analysing, annotating, and interpreting other data. Common types of reference data include: standardised and user defined coding systems, parameters and constants, and units of measure.

Almost all raw data sets undergo processing to apply necessary corrections, calibrations, and transformations. Producing processed data sets may involve filtering operations to remove data that fail to meet the required level of quality or integrity, and data that do not fall into a required specification tolerance. Conversely, it may include merging and aggregation operations with data from other sources. Processing raw data often involves several stages. Therefore the Grid must provide the ability to define workflows, to parallelise operations, and to perform checkpointing and recovery to a point in time in the event of failure.

Result data sets are subsets of one or more databases that match a set of predefined conditions. Typically, a result data set is extracted from a database for the purpose of subjecting it to focused analysis and interpretation. It may be a statistical sample of a very large data resource that cannot feasibly be analysed in its entirety, or it may be a subset of the data with specific characteristics or properties. A copy of result data may be created and retained locally for reasons of performance or availability. The ability to create user defined result sets from one or more databases requires the Grid to provide a great deal of flexibility in defining the conditions on which data will be selected, and in defining the operations that merge and transform data.

Derived data sets are created from other existing processed data, result data, or other derived data. Statistical parameters, summarisations, and aggregations are all types of derived data that are important in describing data, and in analysing trends and correlations. Statistically derived data frequently comprise a significant element of the data held in a data warehouse. Derived data sets can also be created by recording observations inferences, and conclusions that are drawn during analysis and interpretation processes. An important feature of all derived data is that it is often volatile. This is because understanding may change, and hypotheses may be refined over the course of a study. Equally, inference data may not always be definitive, particularly when inferences are drawn about the same corrections and anomalies but are recorded by collaborators with different opinions. For this reason it is important that the Grid provides the ability to maintain personalised versions, and multiple versions of inference data.

### 3.5 Provenance

Provenance, sometimes known as lineage, is a record of the origin and history of a piece of data. It is a special form of audit trail that traces each step in sourcing, moving, and processing data, together with ‘who did what and when’. In science, the need to make use of other worker’s data makes provenance an essential requirement in a Grid environment. It is key to establishing the ownership, quality, reliability and currency of data, particularly during the discovery processes. Provenance also provides information that is necessary for recreating data, and for repeating experiments accurately. Conversely, provenance can avoid time-consuming and resource-intensive processing expended in recreating data.

The structure and content of a record of provenance can be complex because data, particularly derived data, often originates from multiple sources, multi-staged processing, and multiple analysis and interpretation. For example, the provenance of data in an engine fault diagnosis may be based on: technical information from a component specification, predicted failure data from a simulation run from a modelling application, a correlation identified from data mining a data warehouse of historic engine performance, and an engineer’s notes made when inspecting a faulty engine component.

The Grid must provide the capability to record data provenance, and the ability for a user to access the provenance record in order to establish the quality and reliability of data. The mechanisms for capturing provenance should be automated as far as possible to minimise the need for manual entry. The Grid should provide tools to assist owners of existing data to create important provenance elements with the minimum of effort. It should also provide tools to analyse provenance and report on inconsistencies and deficiencies in the provenance record.

### 3.6 Data Access Control

One of the principal aims of the Grid is to make data more accessible. However, there is a need in almost every science discipline to limit access over some of its data. The Grid must provide controls over data access to ensure the confidentiality of the data is maintained, and to prevent users who do have access to the data from changing its content in any way.

In the Grid, it must be possible for a data owner to grant and revoke access permissions to other users, or to delegate this authority to a trusted third party or data custodians. This is a common requirement for data owned or curated by an organisation, e.g. Gene sequences, chemical structures, and many types of survey data.

The facilities that the Grid provides to control access must be very flexible in terms of the combinations of restrictions and the level of granularity that can be specified. The requirements for controlling the granularity of access can range from an entire database down to a sub-set of the data values in a sub-set of the data content. For example, in a clinical study it must be possible to limit access to patients’ treatment records based on diagnosis and age range. It must also be possible to see the age and sex of the patients without knowing their names, or the name of their doctor. The specification of this type of restriction is very similar to specifying data selection criteria and matching rules in data retrieval operations.

The ability to assign any combination of insert, update, and delete privileges to the same level of granularity to which read privilege has been granted is an important requirement. For example, an owner may grant insert access to every collaborator in a team so they can add new data to a shared resource. However, only the team leader may be granted privilege to update or delete data, or to create a new version of the data for release into the public domain.

The Grid must provide the ability to control access based on user role as well as by named individuals. Role based access models are important for collaborative working, when the individual

performing a role may change over time and when several individuals may perform the same role at the same time. Role base access is a standard feature in most DBMSs. It is commonly exploited when the database contains a wide subject content, sub-sets of which are shared by many users with different roles.

For access control to be effective it must be possible to grant and revoke all types of privileges dynamically. It must also be possible to schedule the granting and revoking of privileges to some point in the future, and to impose a time constraint, e.g. an expiry time or date, or a access for a specified period of time. Data owners will be reluctant to grant privileges to others if the access control process is complicated, time consuming, or burdensome. The Grid must provide facilities that, whenever possible, enable access privileges to be granted to user groups declaratively. It must also provide tools that enable owners to review and manage privileges easily, without needing to understand or enter the syntax of the access control specification.

### **3.7 Data Publishing and Discovery**

A principal aim of the Grid is to enable an e-Science environment that promotes and facilitates sharing and collaboration of resources. A major challenge to making data more accessible to other users is the lack of agreed standards for structuring and representing data. There is an equivalent lack of standards for describing published data. This problem is widespread, even in those disciplines where the centralized management and curation of data are well developed. Therefore, it is important that facilities the Grid provides for publishing data are extremely flexible. The Grid should encourage standardization, but enforcing it must not be a pre-requisite for publishing data. It must support the ability to publish all types of data, regardless of volume, internal structure and format. It must also allow users to describe and characterize published data in user-defined formats and terms.

Metadata is the term used for that information which describes a data resource adequately on its discovery. The requirements for metadata are described in more detail in Section 4.6.

The minimum information that a user must know in order to reference a data resource is its name and location. A specification of its internal data structure is required in order to access its content. Knowledge of ownership, currency, and provenance is required in order to establish the quality and reliability of the data content and so make a judgement on its value and use. In addition, specification of the physical characteristics of the data, e.g. volume, number of logical records, and preferred access paths, are necessary in order to access and transport the data efficiently.

### **3.8 Data Publishing Functionality**

It is anticipated that specialised applications may be built specifically to support the data publishing process. Much of the functionality required for defining and maintaining publication specifications is common with that required for defining and maintaining metadata.

The Grid needs to provide the ability to register and deregister data resources dynamically. It should be possible to schedule when these instructions are actioned, and to propagate them to sites holding replicates and copies of the resources. It should also be possible ensure the instructions are carried out when they are sent to sites that are temporarily unavailable. Every opportunity in meeting the requirements must be taken to ensure that, wherever possible, the metadata definition, publication and specification processes are automated and that the burden of manual metadata entry and editing is minimized. There is a need for a set of intelligent tools that can process existing data by interpreting structure and content, extracting relevant metadata information, and populating definitions automatically. In addition, there is need for Grid applications to incorporate these tools into every functional component that interacts with any stage of data lifecycle so that metadata information can be captured automatically.

### 3.9 Data Discovery Functionality

The Grid needs to support data discovery through interactive browsing tools, and from within an application when discovery criteria may be pre-defined. It must be possible to frame the discovery search criteria using user-defined terms and rules, and using defined naming conventions and ontologies. It must also be possible to limit discovery to one or more named registries, or to allow unbounded searching within a Grid environment. When searches are conducted, the Grid should be aware of replicas of registries and data resources, and exploit them appropriately to achieve the required levels of service. When data resources are discovered it must be possible to access the associated metadata and to navigate through provenance records to establish data quality and reliability. It must be possible to interrogate the structure and relationships within an ontology defined to reference the data content, to view the data in terms of an alternative ontology, and to review the data characteristics and additional descriptive information. It must also be possible to examine the contents of data resources by displaying samples, visualizing, or statistically analysing a data sample or the entire data set.

### 3.10 Data Retrieval

The ability to retrieve data within a Grid environment is a universal requirement. Users must be able to retrieve data directly into Grid applications, and into specialised tools used to interrogate, visualise, analyse, and interpret data. The analysis exercise identified the need for a high degree of flexibility and control in specifying the target, the output, and the conditions of the retrieval. These may be summarised as follows:

1. The Grid must provide the ability to translate target, output, and retrieval condition parameters that are expressed in metadata terms into physically addressable data resources and data structures.
2. The Grid must provide the ability to construct search rules and matching criteria in the semantics and syntax of query languages from the parameters that are specified, e.g. object database, relational database, semi-structured data and document query languages. It must also be capable of extracting data from user defined files and documents.
3. When more than one data resource is specified, the Grid must provide the ability to link them together, even if they have different data structures, to produce a single logical target that gives consistent results.
4. When linking data resources, the Grid must provide the ability to use data in one resource as the matching criteria or conditions for retrieving data from another resource, i.e. perform a sub-query. As an example, it should be possible to compare predicted gene sequences in a local database against those defined in a centralised curated repository.
5. The Grid must be able to construct distributed queries when the target data resources are located at different sites, and must be able to support heterogeneous and federated queries when some data resources are accessed through different query languages. The integrated access potentially needs to support retrieval of textual, numeric or image data that match common search criteria and matching conditions. In certain instances, the Grid must have the ability to merge and aggregate data from different resources in order to return a single, logical set of result data. This process may involve temporary storage being allocated for the duration of the retrieval.
6. When the metadata information is available and when additional conditions are specified, the Grid should have the ability to over-ride specified controls and make

decisions on the preferred location and access paths to data, and the preferred retrieval time in order to satisfy service level requirements.

### **3.11 Data Analysis and Interpretation**

Data analysis and interpretation processes may result in existing data being modified, and in new data being created. In both cases, the Grid must provide the ability to capture and record all observations, inferences, and conclusions drawn during these processes. It must also reflect any necessary changes in the associated metadata. For reasons of provenance the Grid must capture the workflow associated with any change in data or in the creation of new data. The level of detail in the workflow should be sufficient to represent an electronic lab book. It should also allow the workflow to be replayed in order to reproduce the analysis steps accurately and to demonstrate the provenance of any derived data.

There are a number of reasons why users may need to carry out analysis on locally maintained copies of data resources. It may be because interactive analysis would otherwise be precluded because network performance is poor, data access paths are slow, or because data resources at remote sites have limited availability. It may be because the analysis is confidential, or it may be because security controls restrict access to remote sites. The Grid must have the capability to record when users signify that they have taken a local, or personal copy of data for analysis and interpretation, and must be able to alert users when the original data content changes. It must also provide facilities for users to consolidate changes made to a personal copy back into the original data. When this action is permitted, the Grid must either resolve any data integrity conflicts automatically, or must alert the user and suspend the consolidation until the conflicts have been resolved manually.

### **3.12 Modes of Working with Data**

The requirements analysis identified two methods of working with data; the traditional approach based on batched work submitted for background processing, and interactive working. Background working is the predominant method for compute intensive operations that process large volumes of data in file structures. Users tend to examine, analyse, and interpret processed data interactively using tools that provide sophisticated visualization techniques, and support concurrent streams of analysis.

The Grid must provide the capability to capture context created between data analyses during batch and interactive workflows, and context created between data of different types and representations drawn from different disciplines. It must also be able to maintain the context over a long period of time, e.g. the duration of a study. This is particularly important in interdisciplinary research, e.g. an ecological study investigating the impact of industrial pollution may create and maintain context between chemical, climatic, soil, species and sociological data.

### **3.13 Data Management Operations**

The prospect of almost unlimited computing resources to create, process, and analyse almost unlimited volumes of data in a Grid ‘on demand’ environment presents a number of significant challenges. Not least is the challenge of managing effectively all data that can be discovered and accessed.

Given the current growth rate in data volumes, potentially billions of data resources of every type and size could be made available in a Grid environment over the next few years. The Grid must provide the capability to manage these data resources across multiple, heterogeneous environments globally, where required on a 24x7x52 hour availability basis. Data management facilities must ensure that data resource catalogues, or registries, are always available and that the definitions they contain are current, accurate, and consistent. This equally applies to the content of data resources that are



logically grouped into virtual databases, or are replicated across remote sites. It may be necessary to replicate data resource catalogues, for performance or fail-over reasons. The facilities must include the ability to perform synchronizations dynamically or to schedule them, and they must be able to cope with failure in the network or failure at a remote site.

An increasing amount of the data in science is being stored in complex data structures of mixed data format and representation. Complex data structures reflect rules and relationships in the data, and this in turn makes the need to maintain referential integrity more critical. A failure in referential integrity can result from media failure, from transaction failure, by failure in synchronization, and by human error. Whatever the cause, it can potentially invalidate part or all of the data. Therefore, the Grid must provide the ability to recover the data to a consistent state at an earlier point in time. This requires that data resources are backed up regularly, and that when necessary recovery can take place online without loss of service.

An increasing amount of data held in complex data structures is volatile, and consequently the potential for loss of referential integrity through data corruption is significantly increased. The Grid must provide facilities that minimize the possibility of data corruption occurring. One obvious way is to enforce access controls stringently to prevent unauthorized users gaining access to data, either through poor security controls in the application or by any illegal means. A second, more relevant approach, is for the Grid to provide a transaction capability that maintains referential integrity by coordinating operations and user concurrency in an orderly manner, as described in [Pearson 02].

## **4 Database Access and Integration Functionalities**

### **4.1 Publication and Discovery**

In a service-based architecture, a service provider publishes a description of a service to a service registry. This registry can then be consulted, by a service requestor, an appropriate service description extracted, and finally a binding created that allows calls to be made to the service by the requestor [Kreger-01]. Such a registry can use standard description models, such as UDDI, or provide alternative project or registry-specific lookups.

The need to provide effective publication and discovery means that descriptions of database services, like other services, must be developed. A basic service description could be the WSDL of the service. Such information is essential to enabling calls to be made to the service, but is likely to be less useful to requestors that want to select a database service based on its capabilities and the data it is making available. Thus it might be useful to publish substantial information on the contents of the database, in addition to details of the operations that the database service supports. The effectiveness of such descriptions would be significantly increased if different services published descriptions of their contents and capabilities using consistent terms and structures. For example, in OGSA [Foster 02a], *service data elements* allow a service to describe itself using an XML Schema. The provision of standard schema definitions for the contents and capabilities of database services seems like an appropriate task for the DAIS Working Group.

### **4.2 Statements**

Database statements allow high-level database operations, such as queries, updates, operation-calls, bulk-loads and schema change operations to be sent to a database system for execution. This implies that the database system over which a service is being provided supports a query or command language interface. This is certainly true for relational databases, but is less uniformly the case for object databases. As such, this is an area in which there may be difficulties supporting consistent service interfaces to different database paradigms and products.

Database statements may involve significant processing time, as they may require access to or transferring of substantial amounts of data. We assume that each operation goes through three phases:

1. Preparation and validation, during which the operation is checked to ensure that it is internally consistent and that it conforms to the data model of the database.
2. Application, during which time updates are performed, or the query evaluated and results constructed.
3. Result Delivery, during which time results are made available to the caller of the operation.

It is envisaged that a statement interface may allow these steps to be performed synchronously or asynchronously. For example, a *Query* operation might perform all three steps above, and return the result, with the service requestor blocking until the query has been evaluated and the result delivered. Alternatively, a *Query* operation may complete after step (1), providing an indication of success to the requestor, which is subsequently informed via a notification mechanism when steps (2) and (3) have completed. Factors such as anticipated query execution times and result sizes could be important in determining which approach is most suitable in a given situation.

A single database service might often support several notations. For example, Xquery and Xpath might both be supported by an interface to an XML repository. When a database service is described in a standard way, it should indicate which notations it is prepared to use, and this information may periodically be modified during the lifetime of the service.

Although standard APIs have been defined for use with multiple database products (e.g., JDBC, ODBC) we know of no standard database statement interfaces within the Web Services community, so this is an area in which the DAIS Working Group might hope to make a contribution. Draft specifications (e.g., [Krause 02]) are available for discussion on the DAIS WWW site.

#### **4.3 Structured Data Transport**

The provision of an efficient data transport infrastructure has been central to Grid middleware from an early stage. Furthermore, work has taken place to provide higher-level services for file access and manipulation, such as GASS [Bester 99]. However, it seems likely that there is scope for additional data transport services that might be used, for example, for delivering large query results. Such a higher-level Grid data transport service might:

1. Deliver data from one source to many destinations along a series of channels.
2. Provide systematic ways to perform transformations, encryption or compression on selected channels.
3. Provide consistent mechanisms for notification of successful or unsuccessful delivery, or for monitoring progress towards completion of a delivery request.
4. Use different protocols for delivery along different channels.

We are not aware of standard service interfaces that cover such capabilities, so such functionalities could be seen as candidates for standardization through the GGF.

#### **4.4 Transactions**

Transactions are crucial to database management, in that they are central to reliability and concurrency control. Transactions, however, are not database-specific artefacts – other programs can and do provide transactional features through middleware services that conform to industry standards (e.g., OMG, J2EE). This section gives an indication of how a transaction service might be

used in conjunction with a database service, and it also explains how such a service may be extended to satisfy a more flexible requirement to coordinate operations across a grid.

A minimal transaction interface that a Grid service might support could include *BeginTransaction*, *Commit* and *Rollback* operations, thereby allowing a client to coordinate accesses to a single database service. However, it may be necessary for a transaction to span multiple services. If a database service is to be able to participate in distributed transactions, it must provide operations for use by the transaction manager that is overseeing the distributed transaction. For example, this might include a *PrepareToCommit* operation for use by a two-phase commit protocol to ensure that all or none of the database services participating in a distributed transaction commit.

Although transactions are a fundamental concept for database operations, the Grid could be felt to require additional and more flexible mechanisms for controlling requests and outcomes than are typically offered by traditional distributed and database transaction models. Some specific differences between the Grid and traditional object transaction environments are:

- Multi-site collaborations that often rely on asynchronous messaging. While this model also occurs in traditional distributed and database systems, the transactions in a traditional system are typically chained together rather than considered as a part of an overall concurrently executing collaboration.
- Operations across the Grid inherently are composed of business processes that span multiple regions of control. Such an environment contrasts significantly with traditional distributed and database systems, where the processing dedicates resources exclusively to the executing transaction (database locks, threads, and so on).
- Traditional distributed and database transaction models optimise execution for high-volume, short-duration transactions and conversations. Grid operations will typically be of longer duration.

Instead of simply extending an existing transaction model to the Grid, an incremental approach may be appropriate:

1. Construction of a core activity service model that provides the capability to specify an operational context for a request (or series of requests), controlling the duration of the activity, and defining the participants engaged in the outcome decision. An example of such a service is the Additional Structuring Mechanisms for the OTS Specification from the OMG [OMG-00], which is also being adopted within the Java Activity Service.
2. Development of a High Level Service (HLS) that provides implementation of patterns typical in a Grid environment, e.g.
  - Traditional distributed and database model where operations occur completely or not at all. Such a completion processing semantic provides the behaviour of a traditional transaction model (i.e., a two-phase commitment semantic).
  - Relaxed semantics such as conversations or collaborations, where a series of operations occur in a more loosely coordinated activity. The participants determine the requirements for completion processing, which may include patterns for compensation, reconciliation, or other styles of collaboration, as required.

Note that the requirement exists to provide a standardized client interface to allow applications to make use of any HLS implementation. A specific proposal for such a client API is outlined in Java, JSR000156 XML Transactioning API for Java (JAXTX),

which is intended to provide a generic API supporting both transactions and extended transactions in a J2EE environment.

The behaviours cited by no means represent a complete list. The relaxed transaction model, however, allows the participating sites to supplement their existing implementations to support more complex processing and relaxed transaction processing models.

It seems to be the case overall, that there is a need for standard interfaces for distributed transaction models for use with Grid services. However, it is likely that transaction standards will emerge from the Web Services community (e.g., XAML), and thus the GGF should perhaps be slow to initiate the development of standards in this area.

#### 4.5 Authentication, Access Control and Accounting

A heterogeneous environment like the Grid necessitates support for different types of authentication mechanism. This is also recognised in the recent security specifications of Web Services [Atkinson 02]. In this section we restrict ourselves to describing specific Authentication, Access Control and Accounting (AAA) requirements for Database Access and Integration, and do not attempt to provide any solutions to the partial list of requirements described below:

- *Delegating Access Rights.* Present day solutions, such as GSI [Butler 00], provide a means of delegating user rights by issuing tickets for access to individual resources. This, however, allows the receiver to impersonate all the user's capabilities. For example, consider a Grid database access mechanism in which Grid-user credentials are mapped onto local-database-user credentials. To provide third party access the user issues a proxy certificate to a user or an application – henceforth referred to as an *impersonator*. Consider a scenario whereby a user has *read*, *update* and *delete* access to a table in a database. Issuing a proxy certificate allows the impersonator to obtain all three access-permissions; restricted delegation can be achieved by using the Community Authorisation Service [Pearlman 01]. A fine-grained delegation mechanism should allow the user to determine the level of access it wishes to delegate. For this purpose both the underlying resource and the Authorisation mechanism should support dynamic partitioning of access rights. For example, consider a user having “execute” rights over a stored procedure that results in the creation of a table or dataset. The stored procedure exists in a database but is owned by the database owner. Execution of the stored procedure results in the creation of the dataset over which the user obtains *read*, *update* and *delete* rights, of which the user wishes to delegate only the *select* rights to a group of service users. This scenario stresses the need for the solution provided to allow dynamic discovery, allocation and delegation of access rights.
- *Authentication mapping for users with multiple roles in the underlying system.* At times, a single authentication identifier/user has multiple roles in the underlying resource, in our case the database. An application performing a particular task should be allowed to authenticate as a “particular” role; for example, JDBC 3.0 connection object allows the user to specify the role in the connection properties. Current schemes for Grid authentication mechanisms do not support role-based access control (RBAC).
- *Lack of accounting methods in current RDBMs and problems with group accounting.* Accounting mechanisms in the Grid aim to provide estimates and resource usage for a given task. Current RDBMs do not typically provide estimates on the approximate time required to complete a given task. Providing such an estimate is difficult as this depends on a number of factors which include and are not limited to: the query in question, the number of databases involved in a distributed query, communication overheads due to

distribution of data, variety of indexing algorithms in use, the caching algorithm, the status of the cache due to previous queries, etc. However, it is absolutely necessary that databases should provide some measure of the actual resources that were used to satisfy a specific request.

The DAIS Working Group is probably not an appropriate place for developing generic AAA standards. However, database services are likely to bring requirements to the fore that are not fully addressed in other settings, and thus there is a need to make explicit the requirements of DAIS and to interact with other standards groups to make them known.

#### 4.6 Metadata

Metadata, the term for describing ‘data about data’, is structured information that adds value to data by conveying context and meaning about the data it describes. Metadata is essential to meeting many other data requirements in the Grid. It is essential for facilitating data management tasks that are involved in maintaining the integrity and consistency of data, and for tasks involved in publishing and discovering data. Metadata is referenced when performing processing, retrieval, analysis and interpretation of data, and it is important for establishing the ownership, currency, validity, and quality of data. Metadata is also essential to the development of Grid services because it enables data operations to be abstracted to a sufficient degree that services can be created and made reusable. This facility makes it possible to access and manipulate data content without knowing where it is physically located, or how it is structured.

Several types of Metadata are important for the development of Grid data access and integration services:

- *Technical metadata* defines the location of data sources and resources; the physical data structure, organisation and grouping of data items into logical records; and those characteristics of the data that are important in deciding how data is best accessed and transported. Technical metadata also defines data currency and history; i.e. versions, and ownership of data.
- *Contextual metadata* defines naming conventions, terminologies and ontologies through which data can be logically referenced. Contextual metadata increases the quality and reliability of data because the definitions conform to agreed syntax and semantics, and also record structural associations and relationships within the data. between definitions, and to define rules for conflicts between mappings.
- *Derived metadata* defines the context and meaning of data derived from any other data. This type of metadata is commonly used in data warehousing environments, when it is often more efficient to store derived data than to recalculate the values dynamically each time they are required.
- *Mapping metadata* defines equivalences between discrete contextual metadata definitions, and between contextual and technical metadata. The ability to map relationships between contextual metadata is particularly important because of the lack of agreed standards in scientific naming conventions for terminologies and ontologies. It enables users to compare classifications and ontologies in terms of their naming conventions, and structural relationships and rules. It also enables them to establish what alternative definitions are available for referencing data content.

The ability to map contextual metadata to physical data structures and schemas is a requirement in order to enable users to access data content using logical references; i.e. without needing to know the definition of its underlying record structure or data schema.

Mapping, in conjunction with contextual metadata, enables users to integrate data sets defined in different classifications and ontologies. This provides the ability to specify a single set of search criteria and data matching rules when performing integrated or federated queries against multiple data resources, and for referencing data in a virtual database.

In terms of standards, most database management standards (e.g., for object and for relational databases) include proposals for the description of various kinds of technical metadata. Many domain-specific coding schemes and terminologies exist with a view to easing the representation of contextual metadata, and the Semantic Web community is developing standards for ontology languages [Heflin 02]. With a view to providing consistent representation of metamodels, the OMG has developed a Meta Object Facility, which is the basis of the Java Metadata interface. However, it is clear that consistent representations of different kinds of metadata are likely to be central to the provision of interoperable DAIS, so it is likely that the DAIS Working Group will propose standard representations for certain kinds of metadata.

#### **4.7 Management: Operation and Performance**

Management of databases in a Grid environment deals with the tasks of creating, maintaining, administering and monitoring databases. In addition, facilities for the management of users and roles and their associated access privileges, resource allocation and co-scheduling, the provision of metadata to client programs, database discovery and access, and performance monitoring tools are all necessary management tasks. Many such high-level management services are currently provided by DBMSs, with each specific system supporting different functionality and presenting a different interface. Most DBMSs are not open source, so integration into a Grid environment will probably require some external wrapping. It is conceivable that in the future, DBMSs will be enhanced to include Grid services. Grid management services must provide the functionality required to integrate existing, autonomously managed databases into a Grid environment, as well as for the creation of new databases. The Grid database administrator should have access to all databases through a common interface.

In a Grid environment, database management services should facilitate both operational management and database management across a set of heterogeneous databases. Operational management includes the creation of roles in a database and the assignment of users and permissions to them. Database management should allow the Grid database administrator to create, delete and modify databases. Authentication and authorisation of Grid based access can be handled by a separate security module that is shared with resources other than databases.

Grid enabled databases will not be exclusively available for Grid based use; access by non Grid users independently of the Grid will probably also be possible. The real database administrator usually has full access privileges, however in a Grid environment, it may be desirable to restrict the privileges of the Grid database administrator for the DBMS. For example, Grid access may only be authorised to certain databases, or there may be a predefined Grid tablespace, outside of which the Grid database administrator has no privileges.

Applications using the Grid to access data will usually need to extract data from multiple database instances, or may need to create new databases across multiple DBMSs that adhere to some application specific requirements (e.g. schema, data types). A priori knowledge of the DBMS (e.g. Oracle, MySQL, DB2) and the schema should not be required. In order to achieve transparency, there is a need for Grid database management middleware that performs the translation of the application's access request into a local database instance request, much like a disk driver translates the user's generic operating system commands into manufacture-specific ones. The Grid database management middleware must expose a common view of the data stored in the schemas of the

underlying database instances. Managing the common Grid layer schema and its mapping to the instance schemas is a nontrivial functionality of Grid database management middleware.

Grid enabled databases will be more prone to denial of service type attacks than independent databases due to the open access nature of the Grid. Current DBMSs can place quotas on CPU consumption, total session time and tablespace sizes. A session can be automatically terminated in the event of excessive CPU consumption, and tablespace quotas can prevent excessive data insertion. Grid database management services must interact with the DBMSs and possibly the underlying hardware to allocate time and resources to queries, and to ensure that ad-hoc Grid based use does not create a denial of service situation.

The Spitfire project of the Data Management Work Package of the EU DataGrid project, [EDG, Spitfire], has already made significant progress in the design and implementation Grid enabled middleware services for access to relational databases. Interfaces have been designed and developed that enable the Grid database administrator to manage databases, tables and user roles. Spitfire only assigns to the Grid database administrator a limited subset of the full privileges available to the real database administrator. The Grid database administrator can only create Grid users of a DBMS, and is not permitted to access databases and tables created outside the assigned Grid tablespace.

As under Statements above, we do not know of standard service interfaces for management tasks. As consistent interfaces to different database systems and models could ease the deployment and maintenance of diverse data management resources on the Grid, consideration needs to be given to the definition of standards in this area within DAIS.

#### **4.8 Data Replication**

Data replication can play a key role for performance and availability of data in the Grid. Replication can be applied to all types of data: databases, files, binaries, etc. Replication may be used: to distribute data for scaling a data intensive workload; to consolidate data from multiple sources, as in a data warehouse for analysis; to cache data from one or more data sources to improve access times, reduce network bandwidth requirement and provide better availability; to checkout data from one or more sources for data analysis or collaboration; to make replicas for disaster recovery; or to clone binaries, e.g., OS, middleware or applications.

In the Grid, a replication service may be invoked dynamically by schedulers or workload managers to perform just-in-time replication in concert with the allocation of compute resources. The notion of replication quality of service (QoS) is defined as the latency of end-to-end data propagation from the source to the target from an application's standpoint. Examples of end-to-end latency are: (i) the time between when updates are committed on the source, and subsequently those updates are committed at the target; and (ii) the time between when query results are produced and are committed on the target. Factors involved in end-to-end latency are: capturing updates at the source, processing to buffer data on the network channel, network latency, queuing, transformations, applying of updates at the target, etc. A replication service may monitor QoS and may interact with resource managers to meet user-specified goals.

It is important to track replicas [Foster 02b]. For example, schedulers may choose to schedule tasks on the nodes where the required data already exists and satisfies currency requirements. The database optimizer may take advantage of replicas to speed up a query by performing local data access. Note that for database exploitation it is also important to track the schema version when a replica is made, and the currency of a replica.

We now describe what current database products [DB2, Oracle] provide, and what they lack for the Grid environment. They provide support user specification of the source, target, filters and transformations to be performed during replication.

A *Capture* program on the source:

- *captures* updates by reading the database log so that it is non-intrusive to the workload,
- *filters* and subsets the columns that need to be sent to the target, and
- *sends* updates to the *Apply* program on the target using a point-to-point channel or a staging area.

An *Apply* program applies the updates logically (e.g., via SQL interface) at the target, auditing the subscription. Detection of failure and recovery is the responsibility of the Capture and Apply programs themselves.

Recent advances in DBMS, whereby one or more columns of an SQL table can have file references [Michel 01, Melton 00], allow replication of files and their associated data (in the database) transactionally. A column can be annotated to indicate that it references a file. The replication source sends this annotation and the column value, which is typically the url of the file. As the Apply program notices the annotation, it first replicates the file with possible file name transformation, modifies the column value with the transformed filename, and then updates the target database.

For the Grid, current approaches could be enhanced considerably, from both engineering and functional standpoints. For example, effective function componentization would simplify development of the Capture and Apply programs. Functional extensions could include: dynamic subscription for just-in-time replication; specification of QoS requirements; support for replica catalogues; support for replication of files and associated metadata in the database with consistency; and flow control that handles variable target speeds, etc.

With federated database technology [Haas 02, Melton 00], query results from several relational and non-relational data sources can be joined, aggregated or reshaped and then applied to one of several relational databases at the target. For details of a proposed Data Replication service for the Grid, see [Gavacs 02].

In terms of service interfaces, there is a requirement for interface definitions that allow consistent creation, description and maintenance of replicated data on the Grid. The GGF seems like a suitable forum for the standardisation of such interfaces. The DAIS group can perhaps help to inform discussions in the Data Replication Research Group, by bringing a database perspective.

## 4.9 Connections and Sessions

The aim of this section is to describe connections and sessions in the context of DAI services, and also to describe various options that determine the lifetime of connections and sessions. This is considered relevant to the document as a whole, in that designers of DAIS need to consider the role and nature of services and connections in service definitions.

It is assumed that a single service instance can expose a number of database instances. A state-full/stateless interaction with a service constitutes a “session”. Establishing a communication with the service initiates a session with the service. The session cannot be explicitly shared between different users, while a user can establish multiple sessions with a service. All the subsequent interactions between the user and the service are within the scope of a particular session. However, a session can outlive the lifetime of the individual communication links between the user and the service.

In the case of Grids, it may be desired to regain access to a previous session, but we propose that such logic should be specific to the service, as regaining a session may have security implications on some authentication/authorisation mechanisms.



A connection refers to a communication with a specific database instance. A connection is always established under the context of the current user session. A session may hold a number of connections. A connection's lifetime is limited to that of the session containing it. The connection context may support transaction management, but the implementation should determine whether a connection can/cannot participate in a transaction.

#### 4.10 Integration

One of the aims of the Grid is to promote the open publication of scientific data. If this is realized then it is expected that many of the advances to flow from the Grid will be from applications that combine information from multiple data sets. An investigation into the requirements of early Grid projects has concluded that the prospect exists for millions of data resources and petabytes of data being accessible in a Grid environment [Pearson 02]. Support for federating data resources is therefore vital to the success of the Grid – the alternative of forcing each application to interface directly to a set of databases and resolve federation problems internally would lead to application complexity, and duplication of effort.

Database federation has been comprehensively studied [Sheth 90], and implementations are available from vendors. However, these are unlikely to meet the needs of all Grid applications – the factors that make Grid database federation different include:

- *Highly dynamic federations.* On the Grid, many applications will dynamically select a set of databases to access based on the results of queries of metadata catalogues. This, and the fact that the set may be very large, places a premium on fully automatic federation without user input. This needs resolution at the interface level (how can the federation middleware communicate with the different interfaces offered by the databases to be federated?) and the semantic level (how is data to be interpreted so that it can be integrated?).
- *Extreme performance.* A query that joins data from a set of very large databases could have huge network, CPU and disk IO requirements. Therefore, a scalable performance solution is required, and the Grid offers opportunities for providing it: Grid-resources can be dynamically acquired in order to meet the performance requirements of the query, while the application of parallel query execution techniques [Smith 02] across the Grid could provide scalability.
- *Alternative source selection.* Replica management systems will provide alternate sources of the same data. The selection can be made on the basis of availability (e.g., in the presence of failure), performance and cost.
- *Exploiting the Semantic Grid.* The aim of the Semantic Grid [De-Roure 01] is to provide metadata that assists in the machine interpretation of data. This can be exploited in schema integration, e.g. in identifying sets of relational columns whose data has the same interpretation.
- *Use of Grid standards and services.* Federation middleware will have to conform to the emerging OGSA if it is to be seamlessly integrated with other Grid services. This will ensure a standard security interface, and offer opportunities to exploit other appropriate services such as those for efficiently transferring large amounts of data. The role of Grid standards becomes of prime importance if the approach advocated in this paper – wrapping DBS so that they offer standard, OGSA conformant database services – is taken. If this is the case, then the federation middleware can interrogate the service metadata to determine what level of interfaces each database provides (e.g., the query language supported) and, if possible, generate a new single, “virtual” database service interface that represents their federation. This can be done on a service-by-service basis – while federated query services are likely to

be the most important, a whole range of other services offered by databases will need to be federated for some applications, including transactions, metadata, notification, accounting and scheduling.

## 5 Conclusions

This document has provided an overview of requirements and functionalities for Grid Database Access and Integration Services. It is not claimed that the requirements are complete. Nor is it claimed that the functionalities described are: (i) sufficient to address all the requirements (they are clearly not); or (ii) partitioned in a way that directly reflects how they should be standardized.

There are several important issues that must be addressed within the DAIS Working Group to enable standards to be brought forward through the GGF, which it is hoped this document will help to address. These include:

1. *Priorities*: what functionalities of relevance to DAIS are all of: (i) widely recognised as important; (ii) reasonably well understood; and (iii) not the subject of existing standardization activities elsewhere. Such functionalities are candidates for standardization within the DAIS Working Group or elsewhere in the GGF.
2. *Scope*: what functionalities of relevance to DAIS are within the remit of the DAIS Working Group? As database access and integration services overlap with many other areas (security, replication, data transport, etc), certain issues are likely to be addressed in interactions across several GGF groups.
3. *Granularity*: how much ground should be covered by any one proposal for a standard. For example, there are clear relationships between statements and transactions, but equally much can be said about each of these functionalities that is orthogonal to the other. While specific proposals for standards are likely to have champions, it will be good if the community is able to reach broad agreement on the nature and scope of individual proposals early in their development.

## 6 References

- B. Atkinson, et al., Web Services Security (WS-Security), Version 1.0, Available from: <http://www-106.ibm.com/developerworks/library/ws-secure/>, 2002.
- J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke, GASS: A Data Movement and Access Service for Wide Area Computing Systems, 6<sup>th</sup> Workshop on I/O in Parallel and Distributed Systems, 1999.
- R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer and V. Welch, IEEE Computer, 33(12), 60-66, 2000.
- DB2 Universal Database, Replication Guide and Reference, IBM, Available From: <http://www-3.ibm.com/software/data/dpropr/library.html>.
- D. De-Roure, N. Jennings, N. Shadbolt and M. Baker, Research Agenda for the Semantic Grid: A Future e-Science Infrastructure, [www.semanticgrid.org](http://www.semanticgrid.org), 2001.
- EDG Grid Data Management <http://cern.ch/Grid-data-management/>.
- I. Foster, C. Kesselman, J. Nick, S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, <http://www.globus.org/ogsa/>, 2002a.
- I. Foster et al, Giggle: A Framework for Constructing Scalable Replica Location Services, 2002b.

N.J. Gavacs, S. Glanville, S. R. Jeffery, J.P. Mcalister, I. Narang, V. Raman, Data Replication Service in Grid, IBM Technical Report (in progress), 2002.

A. Krause, K. Smyllie and R. Baxter, Grid Data Service Specification for XML Databases, Edinburgh Parallel Computer Centre Report: EPCC-GDS-WP3-GXDS 1.0, Version 1.0, 2002.

H. Kreger, Web Services Conceptual Architecture, Technical Report WCSA 1.0, IBM Software Group, 2001.

L. Haas, E. Lin, M. Roth, Information Integration through Database Federation, to appear in IBM Systems Journal, November 2002.

J. Heflin, R. Volz and J. Dale, Requirements for a Web Ontology Language, W3C Working Draft, <http://www.w3c.org/TR/webont-req>, March, 2002.

R. Michel, et al., Data Links Managing Files using DB2, <http://www.ibm.com/redbooks>, 2001.

J. Melton et al, SQL and Management of External Data. Also ISO/IEC 9075-9-2000, Information Technology – Database Languages – SQL – Part 9: Management of External data (SQL/MED).

OMG – Additional Structuring Mechanisms for the OTS Specification. Technical Report ORBOS/2000-04-02, Object Management Group, 2000.

N.W. Paton, M.P. Atkinson, V. Dialani, D. Pearson, T. Storey and P. Watson, Database Access and Integration Services on the Grid, UK e-Science Programme Technical Report Number UkeS-2002-01, 2002.

Oracle Streams, Technical White Paper, Available from: <http://technet.oracle.com/products/dataint/content.html>, June 2002

L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke, A Community Authorization Service for Group Collaboration, Submitted to 3<sup>rd</sup> Intl. Workshop on Policies for Distributed Systems and Networks, 2001.

D. Pearson, Data Requirements for the Grid: Scoping Study Report, Presented at GGF4, Toronto, <http://www.cs.man.ac.uk/grid-db>, 2002.

A.P. Sheth and J.A. Larson, Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, ACM Computing Surveys, 22(3), 183-236, 1990.

J. Smith, T. Gounaris, P. Watson, N.W. Paton, A.A.A. Fernandes and R. Sakalleriou, Distributed Query Processing on the Grid, Available from: <http://www.cs.man.ac.uk/grid-db/>, 2002.

Spitfire project: <http://cern.ch/hep-proj-spitfire>.