Grid Data Services – Relational Database Management Systems – Version 1.0 – 05Jul2002

Authors

Brian Collins – collinsb@uk.ibm.com (1)
Andrew Borley – borley@uk.ibm.com (1)
Neil Hardman – nhardman@uk.ibm.com (1)
Alan Knox – knox@uk.ibm.com (1)
Simon Laws – simon_laws@uk.ibm.com (1)
James Magowan – magowan@uk.ibm.com (1)
Manfred Oevers – manfred_oevers@uk.ibm.com (1)
Ed Zaluska – zaluska@uk.ibm.com (2)

Addresses

(1) IBM United Kingdom Laboratories
Hursley Park
Winchester
SO21 2JN
United Kingdom

(2) Dept. of Electronics and Computer Science
University of Southampton
Southampton
SO17 1BJ
United Kingdom

Notices

Copyright

Versions

Date	Version	Changes	Status	Approver
05Jul2002	1.0	First Version (Working Draft)	Unresticted Public	Matrix Project Manager

Distribution List

Version	Organisation	Distribution
1.0	GGF	DAIS Working Group

Abstract

This document is for information and should be viewed as a working draft. It describes our current thinking on Grid Data Services for access to Relational Database Management Systems (RDBMs).

The objective of this document is to allow critical validation of our approach and proposals in the wider Grid community through the GGF DAIS Working Group. We plan to revise this document for GGF6 based on input from the community.

We start with some background to our proposal and describe the current scope and activities we are undertaking and put those into the context of other Grid activities, which are closely related to ours. We then describe our approach, which fundamentally begins with gaining an understanding of the user scenarios, and the generic patterns that arise from those. We then describe our current thinking on interface definitions and some of the choices with which we are faced. We finally describe the implementation activities we are undertaking, in parallel with the scenarios work, through prototyping experiments.

The thinking of the Grid and Web Services communities are converging, as evidenced by OGSA, however decisions have yet to be made in several key areas, for example security, stateful services, and service discovery. This means that we have to be aware that the environment within which we are working is evolving and that we will have to adapt our proposals accordingly.

Background

Relationship to other Activities

The *Open Grid Services Architecture – Data Access and Integration Services* project (known as OGSA – DAI) [1] is an initiative of the UK National eScience Centre[5], [6], and [7].

In this project the study of Grid data requirements [9] has been divided up into work packages [1], [10], and **Error! Reference source not found.**. Work package 6 (hereafter referred to as Matrix) deals with the integration of RDMS into Globus 2. However, the aim of the OGSA-DAI project in general is to provide an overarching framework that integrates access to RDBMS with access to XML as Grid Data Services under OGSA [3] and [4]. The architecture and design of Matrix must therefore be considered in the context of the ongoing work on OGSA and, more importantly, the products of the other OGSA-DAI work packages. In this way Matrix can define a migration path and protect the investment of those Grid projects current committed to Globus 2, but who may consider moving to OGSA once a stable release is available.

Our work has been progressing in parallel with the Architecture activities of Work Package 2 [47]. It is noted that already there is much common thinking between the Architecture and our current proposal even though they were initially developed independently. The longer-term objective is to rationalise any differences, to ensure that Matrix is viewed as a “plug-in” to the overall Architecture, and to ensure consistency of interface and operation when compared to the other OGSA-DAI work packages [28], [32], [33], [34], and [47], other related Data Access projects [23], [24], [25], [26], and [36], and other RDBMs proposals [15] and [35].

The primary sources used in the generation of this document were Technical Reports from Globus [2] and [3]; a proposal for supporting Databases on the GRID [8]; a detailed requirements analysis [9]; initial scope and functionality proposals [10], [11], and [32]; and an initial architecture and design proposal [12]. In addition inputs from the following technical documents and reviews on OGSA [4], [13], and [14], and Query Languages [18], [19], [20], [29], [30], and [31], Transformations [44], SSL [45], and Web Services Security [46] were also used.

Scope and Objectives

The goals of the whole OGSA – DAI project as detailed in [1] are fourfold:

1. To produce a recommended specification for data access and integration services for OGSA.
2. To produce reference implementations of OGSA services for accessing XML data repositories and relational databases.
3. To produce a recommended specification for additional OGSA services required by the e-Science communities.
4. To produce data integration middleware which allows access to existing forms of data repository through Globus-2 (with compatibility for Globus-3).

Matrix is divided into two main phases

Phase 1 – Core Grid Services Prototype

The aim of this phase is to provide a prototype implementation to allow users of the Grid to access existing data sources through the Globus-2 API. However, it is the intention that the prototypes produced by this phase will be compatible with both Globus-2 and the emerging Globus-3. We believe that this compatibility will be crucial as applications migrate between the Globus versions.

This work will only support RDBMs which are enabled for JDBC [16], [17], and [27] and testing will be limited initially to MySQL, Oracle and DB2 and which, at minimum, support the SQL92 standard. This work will not support integration across multiple, heterogeneous RDBMs, but it could support access to an already federated environment through JDBC. The Globus-2 integration will be limited to the Globus Security Infrastructure (for access and authentication) and to GridFTP (for data delivery).

During this phase many simplifying assumptions will be made that would prevent widespread deployment of the prototype – the aim is to generate a “proof-of-concept”, not a widely used middleware service.

During this phase software will be developed on the assumption that the code may not persist for use in later versions. The prototype will be developed on the assumption that it will be deployed for the purpose of evaluating the performance of the functionality provided. The following is what we currently propose will constitute this Phase:

1. The publication and review within the OGSA-DAI project of a proposed architecture and design for a set of Core Grid Services to support access to RDBMs. The Core Grid Services will be designed to:
   - support any RDBMs which is enabled for JDBC.
   - provide stateful services.
   - provide high performance delivery of large datasets using a number of protocols (for example, SOAP/HTTP and GridFTP).
   - conform to the OGSA specification.
   - abstract the OGSA specification to allow Globus-2 integration
2. The development, testing and documentation of a prototype implementation of the set of Core Grid Services which will:
   - be based upon Globus-2 (specifically using GSI and GridFTP).
   - be implemented by abstracting the OGSA/Globus-3 specification to allow both Globus-2 integration and to facilitate migration to Globus-3/OGSA.
   - provide a Java API for application developers
   - be available for testing with Oracle, DB2, and MySQL.

   The Core Grid Services will not be optimised to provide high performance delivery of large datasets during this phase (see Phase 2 – Full Grid Services Reference Implementation below).
3. Evaluation by early adopter Grid projects

We would envisage that the deliverables from this phase would be superseded by the deliverables from the subsequent twelve months phase. However, all attempts would be made to provide a migration path for any early adopter applications to the next version.

Phase 2 – Full Grid Services Reference Implementation

The aim of this phase is to provide reference implementations of the prototypes which were developed in Phase 1. However, a revised Scope and Objectives for Phase 2 will be one of the deliverables from Phase 1. A key input to that will be the review and evaluation through the GGF DAIS Working Group and especially by the early adopter projects. The following is the current view of what might constitute this phase.

1. Continuing research and information gathering on relevant activities within Grid and related areas such as Web Services.
2. Publication and external review of a reference architecture and design for a set of Full Grid Services to support access to RDBMs. The Full Grid Services will be designed to:
   - support any RDBMs which is enabled for JDBC.
   - provide high performance delivery of large datasets using a number of protocols (for example, SOAP/HTTP and GridFTP).
   - progress towards a Grid Services Standard through the GGF DAIS Working Group.
3. Development, testing and documentation of a reference implementation of this set of Full Grid Services will

- be based upon Globus-3 (specifically utilising GSI and GridFTP) with backward compatibility with the prototype Globus-2 implementation (see Phase 1 – Core Grid Services Prototype above) where possible.
- provide a Java and possibly a C API for application developers.
- be tested with Oracle, DB2, and MySQL.
- will be optimised to provide high performance delivery of large datasets using a number of protocols (for example, SOAP/HTTP and GridFTP)
- be evaluated by early adopter Grid projects.

Introduction

This paper discusses 'Matrix', a middleware project which plans to provide comprehensive and efficient database access in a Grid environment.

The recent widespread interest in Grid computing [2] has identified several research areas of considerable interest to both the e-Science community and also more generally the wider e-Business marketplace. One such research area is the Grid-enabled access of Relational Database Management Systems (RDBMS), with particular emphasis on two different modes of access:

- The ability to reference very large databases and very large data items.
- The ability to reference a number of geographically distributed databases in a single Virtual Organisation [2].

Much of the existing Grid work to date has, from necessity, incorporated RDBMS facilities on an ad hoc basis, rather than establishing a coherent environment that would be applicable over a range of projects. The EU-funded DataGrid project [36] plans to implement such an access environment, albeit with a restricted range of facilities.

The work reported concentrates on the requirements analysis of the necessary infrastructure, with an emphasis on deriving a general solution which will be applicable both to current Grid systems based on Globus Toolkit 2 and also the future Open Grid Service Infrastructure/Interfaces (OGSI) under development by the OGSI working group of the Global Grid Forum [37]. OGSI has developed out of the Open Grid Services Architecture (OGSA) specification [3].

Approach

This paper documents a work in progress. As we have taken a top down approach to this task we have focused our efforts to date on identifying the requirements for RDBMS access in the Grid in the form of abstract scenarios. In order to understand the implications of these requirements there is an ongoing prototyping exercise. The ultimate deliverable is the specification of the interfaces that a user of the service will use. Thus the approach can be visualized as in Figure 1.

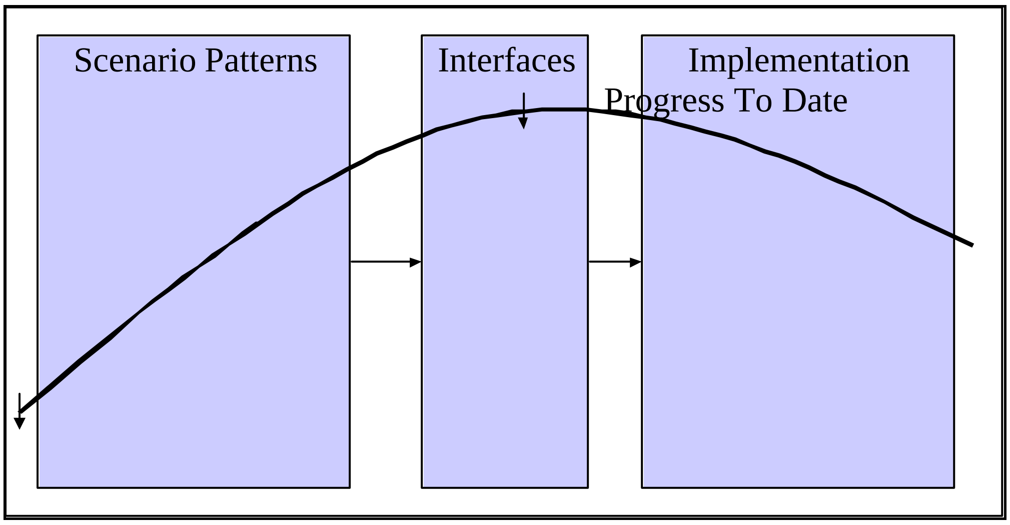

Figure 1 – Top down approach

In this figure the curved progress line demonstrates that working from the top the focus has been on capturing scenarios and on considering their implications through prototype implementations.

Scenarios and Patterns

Requirements

Understandably, there are many sources of requirements for this project. There are many e-Science projects that rely on data sharing and hence data access in the grid environment, for example, AstroGrid [23] and MyGrid [21]. The European DataGrid project [36] is also facing these problems and has made significant progress in developing solutions. All of these activities provide requirements to this project.

Following the major requirements study by Dave Pearson [9], we have chosen to enumerate requirements in terms of the abstract scenario patterns that the software must support. The scenario patterns considered in this design are described in more detail in a separate document [38].

In the first stages of our work we are concerned primarily with supporting query and metadata operations on remote databases. On the face of it this sounds like straightforward database access using existing techniques. However this gives us the platform to progress to more complex scenarios and address the isolation and subsequent integration of query and delivery services.

This is important as the project must support and satisfy wider architectural criteria, i.e. the need to support different source data types, data replication, data federation etc. In particular we must deliver an attractive proposition for users of the Grid. We must provide a solution that can be used in the simplest scenarios with little effort required in installation and configuration. It should also provide a framework that allows “power” users to tune and extend its modes of operation.

Scenario Capture Process

The process of identifying the scenarios that form our requirements statement is iterative.

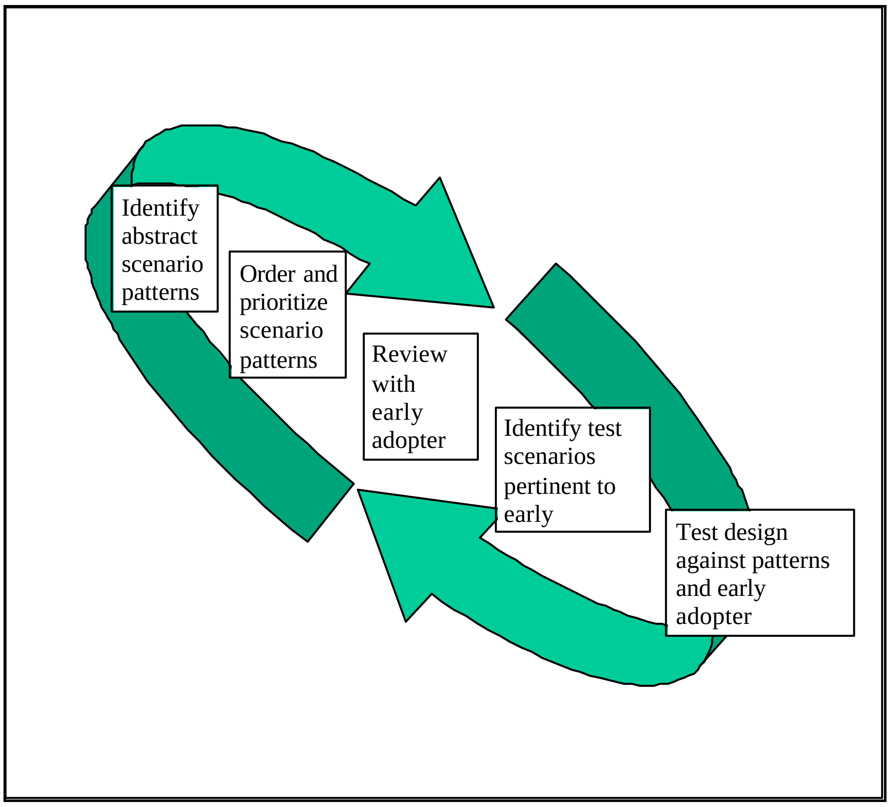
Identify abstract scenario patterns
Order and prioritize scenario patterns
Review with early adopter
Identify test scenarios pertinent to early
Test design against patterns and early adopter

Figure 2 – The scenario capture process

In this process Scenarios represent real world application-specific descriptions of RDBMS access mechanisms. Scenario patterns are abstract patterns for data access which are derived from the real world scenarios but which are not themselves application specific.

The scenario patterns identified to date have been categorized using a scheme based on that proposed in [8].

Vertical	**Horizontal**
Query – Retrieve (R)	Meta Data (M)
Query – Update (U)	Transformation (T)
Bulk Load (B)	Notification (N)
Data Management (DM)	Scheduling (S)
High Level (H)	Accounting (A)
	Security (SEC)
	Error (E)

Figure 3 – Scenario Pattern Categorization

“Vertical” and “horizontal” in figure 3 distinguish fundamental scenario patterns (vertical) from more general support patterns (horizontal) that could apply to several of the vertical patterns.

In total 42 scenario patterns have been identified to date covering all of the categories in figure 3. Many of these scenarios identify distinct modes of operation, while others simply represent variations on a theme.

The key actors in the patterns are *Matrix*, *Analyst* and *Consumer*. These roles account for patterns where some query against a data source is constructed and refined by some party (Analyst) with the intention that the results of the final query are sent to a third party (Consumer) for processing. Scenario patterns where the Analyst and the Consumer are one and the same are essentially a special case.

Given below is a selection of example scenario patterns (identified R1, R2, R10 and R21) taken from our initial studies that provide the most important modes of operation that we are currently considering.

Scenario Pattern R1

“Select a small number of rows each with a small volume of data returned directly to the calling process”

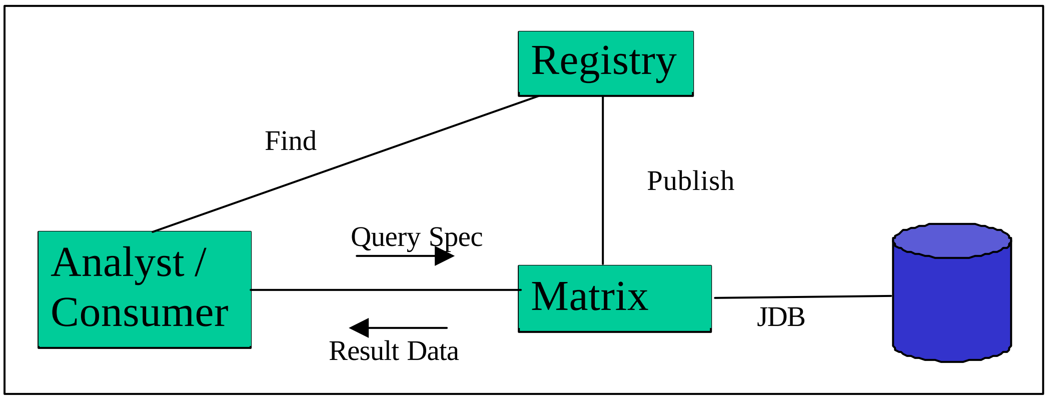

Figure 4 – Scenario Pattern 1

Some mechanism is assumed to publish the Matrix service to a registry, which can then be searched by the user to discover how to access the correct service. The user (shown as 'Analyst/Consumer' in the diagram, because in this scenario pattern these two functions refer to a single user) issues a query specification to the Matrix infrastructure, which interrogates the RDBMS and returns the resulting data directly to the user. The same model also supports several of the metadata scenarios described in the scenario patterns document [38], returning metadata rather than the actual data.

Scenario Pattern R2

“Select from a database table that contains references to external information. This is often the case in scientific Grid applications where the database tables hold metadata about images or other data sets held externally to the database. The external reference is returned as part of the resulting information.”

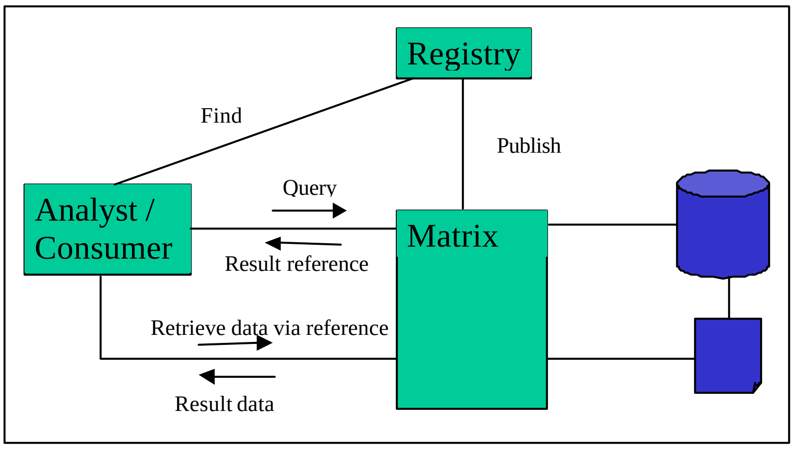

Figure 5 – Scenario Pattern R2

In this scenario pattern, the RBDMS returns a reference to the actual data. The Analyst may access this data using the returned reference via Matrix. Direct access to the data by the Analyst to the data is not considered.

Scenario Pattern R10

“Third party or parallel transfer. A select with a large volume of data to be returned indirectly, i.e. to a Consumer, which could be a file system, replica manager or another service. The Consumer is given authority to access the retrieved data set and the original requester is provided with a reference to the delivered data set.”

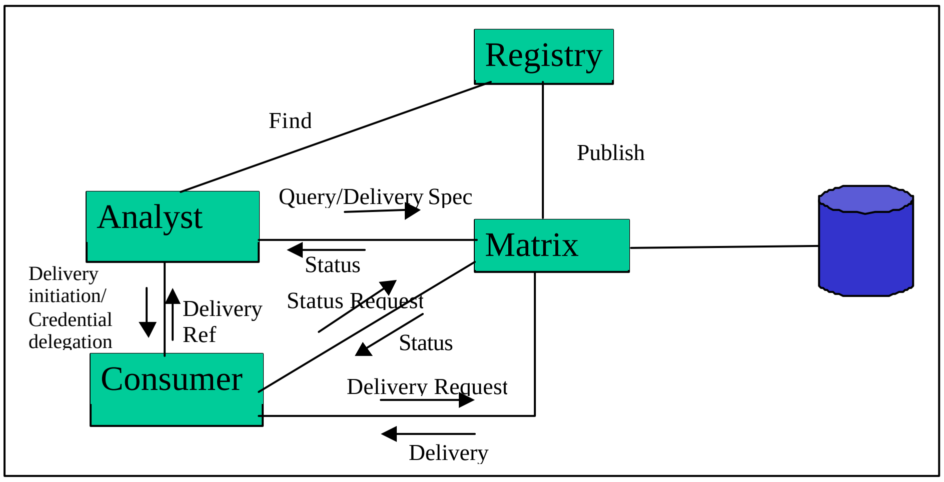

Figure 6 – Scenario Pattern R10

In this scenario pattern, the user function is split into 'Analyst' (the query specifier) and 'Consumer' (the data user). The Analyst may need to obtain a reference from the Consumer before the query can be sent (together with the delivery specification) to Matrix. The result returned to the Analyst is the status of the request. When the data becomes available, the Analyst can notify the Consumer, which can then request delivery (monitoring progress with explicit status request messages). Alternatively, the Consumer could be designed to monitor the status of the initial query from the outset.

Scenario Pattern R21

"Return metadata directly to the user, but deliver the result set to a third party. (Note – In the AstroGrid project the most common mode of usage is expected to be data placed on an FTP server.)"

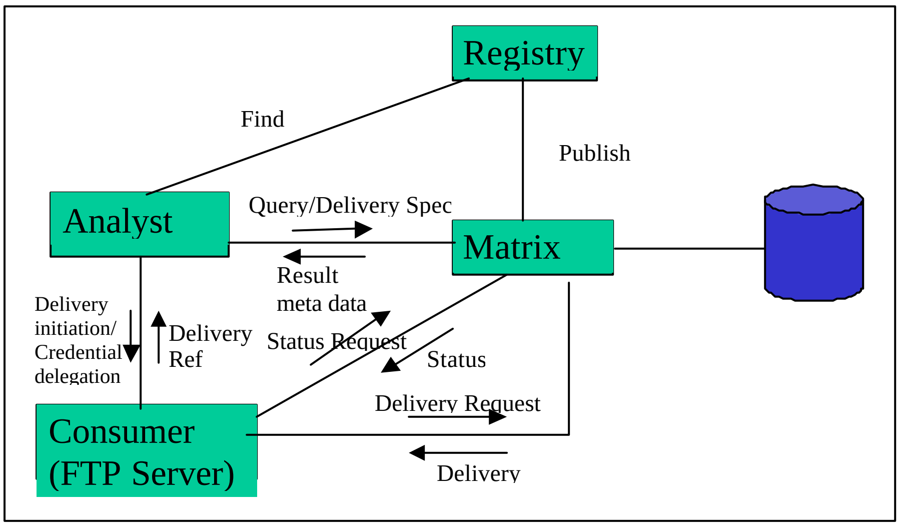

Figure 7 – Scenario Pattern R21

The essential difference between this scenario pattern and R10 is that the metadata (rather than just the status) is returned to the user, who will then access results as required directly from the FTP server.

Other Scenarios

In addition to the examples given above, the initial analysis demonstrated that there is a wide range of potential requirements to be satisfied - for example:

- Translation of query or results.
- Control over maximum size of result set or query execution time.
- Selection of alternative delivery mechanisms (e.g. stream or FTP).

These are documented in the scenario patterns document [38].

Initial Matrix functions

Analysis of the scenarios we have collected has led us to consider the Matrix service as comprised of three groups of functions.

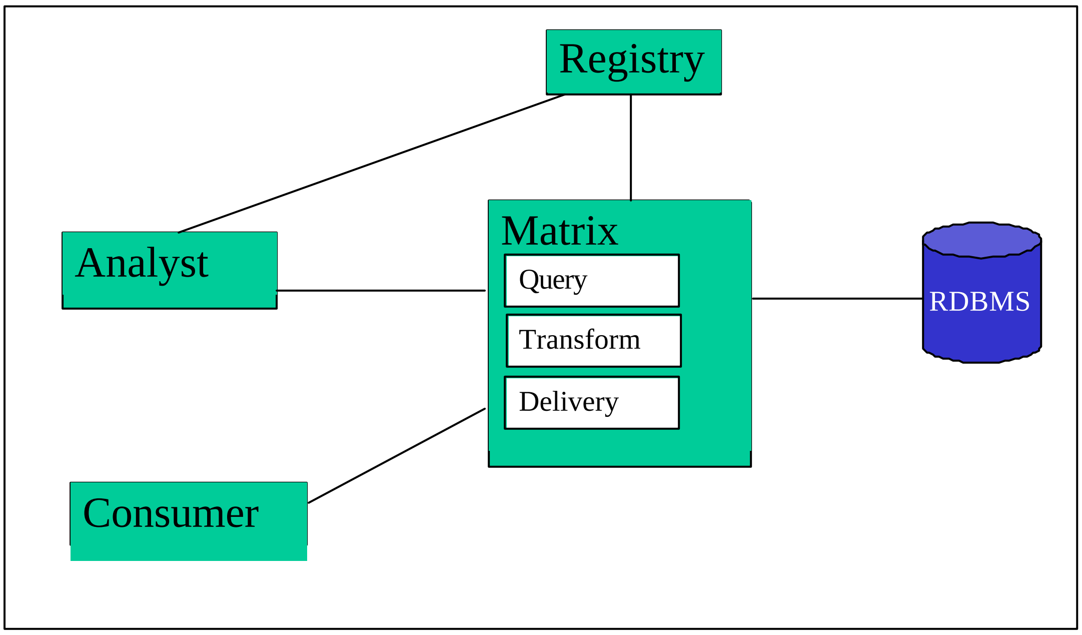

Figure 8 – Key functions

The function of Matrix splits up into three areas:

- Query – This relates to the execution of SQL against a relational database.
- Transform – Conversion of the retrieved data into a new format. Examples of transforms might be data compression and application level encryption.
- Delivery – Movement of the retrieved data to one or more Consumers.

These functions are referred to later in this paper as QTD.

Interfaces

Discovery

As yet we have not defined the Matrix metadata that will be published to ensure efficient discovery. It is unclear which of many registries will be used and this aspect will be addressed later in the project.

Security

The options for securing a Matrix Service at the Transport level are Secure Sockets Layer (SSL) and Grid Security Infrastructure (GSI) [39]. Web Services Security [40] offers the option of application level security. Each of these is capable of meeting the authentication (the user must have valid Grid credentials), confidentiality (security of data in transit) and integrity (the data received by the user must be what was sent by the database) requirements for the service.

There is an additional requirement of authorization: a user with a valid Grid credential must be authorized to access the target database in some role. This requirement is addressed by mapping Grid credentials to user IDs in the target database. This operation can either be performed explicitly within the Matrix system by maintaining a suitable map at the Matrix node, or can be implicit with the system by making use of the mapping of Grid credentials to user IDs provided by GSI in the file */etc/grid-*

security/grid-mapfile. The implications of a single instance of the Matrix service managing multiple databases needs to be considered carefully before adopting the latter approach.

The Spitfire (European DataGrid Work Package 2) project [36] has implemented a security architecture based on transport-level SSL security and mapping of Grid credentials to database roles. This implementation has been developed with reuse in mind and could be applied for Matrix.

State

A stateful service allows the results of a query against the database to be cached close to the database for subsequent queries or delivery (possibly delayed) to a third party. This will be a requirement for example if it is important that the data delivered is identical in every respect to the data queried. The implication here is that the database node must manage a number of sets of results for an indefinite time. Both the number and size of result sets may be large and the scalability of this solution is questionable. The alternative is to leave data in the relational database and accept that updates may occur between query time and delivery time.

There are three approaches to resolving the issue of state:

- Stateless – We ensure through the design of our interfaces that there is never any need to hold state.
- Non-OGSA Stateful – Here we do not assume the existence of any Grid toolkit implementing OGSA and we address the issue of state ourselves
- OGSA Stateful – We make use of a Grid Service Handle (GSH) and allow the OGSA compliant infrastructure to manage the state for us.

We see Matrix as evolving from Stateless to Non-OGSA Stateful to OGSA Stateful over time, as Globus Toolkit 3 (OGSA implementation) becomes more widely adopted and replaces Globus Toolkit 2.

Below we consider state using the example where we have an Analyst who wishes to make a query and have the results of this query returned to a separate Consumer.

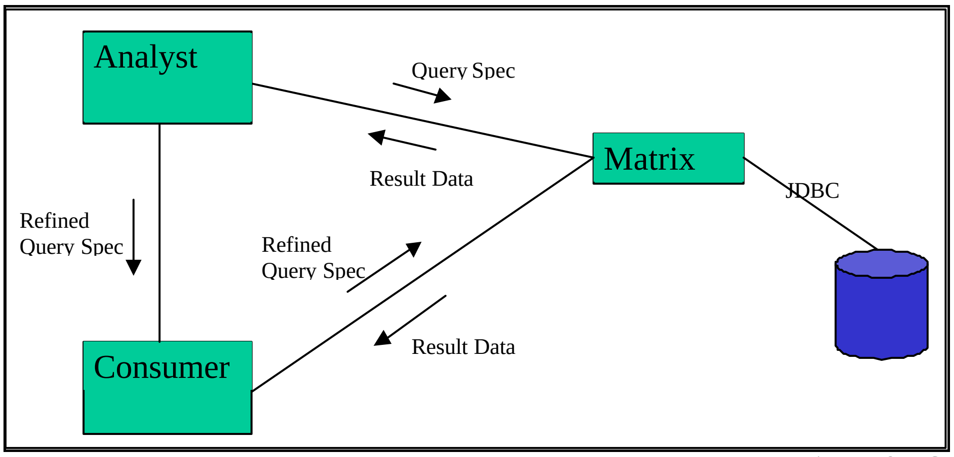

Figure 9 - Stateless

With a stateless approach the Analyst would send the delivery specification to Matrix, which carries out the query and returns the results to the Analyst who then forwards them to the Consumer. This is not ideal especially if the results are large but could be improved by sending a query specification to Matrix requesting metadata to be returned to the Analyst. At this point query refinement can take place until a satisfactory query has been developed, which can then be sent to the Consumer. The Consumer can issue the new query to Matrix, which executes the query and returns the results.

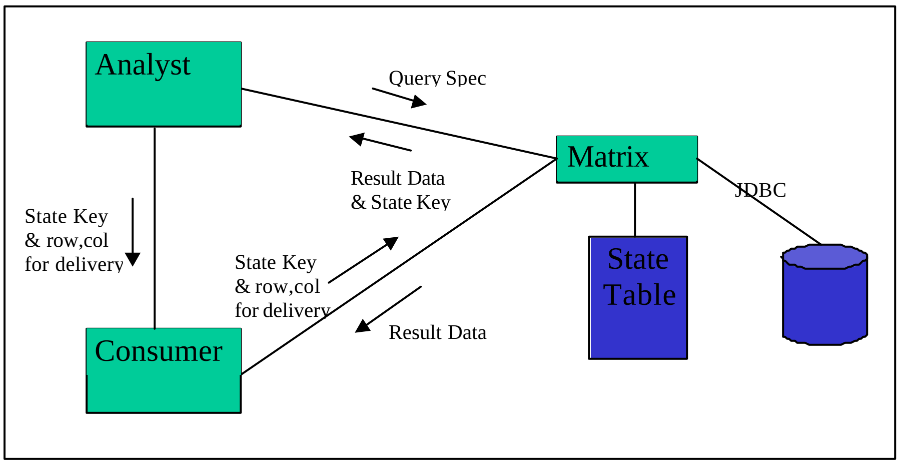

Figure 10 – Non-OGSA Stateful

In a Non-OGSA Stateful, Matrix would maintain a state table accessed using a state key. The query specification sent by the Analyst results in a state key as well as any data requested for return. The state key can then be passed to the Consumer with an indication of the data sought. The Consumer can access Matrix using the state key and obtain the data required.

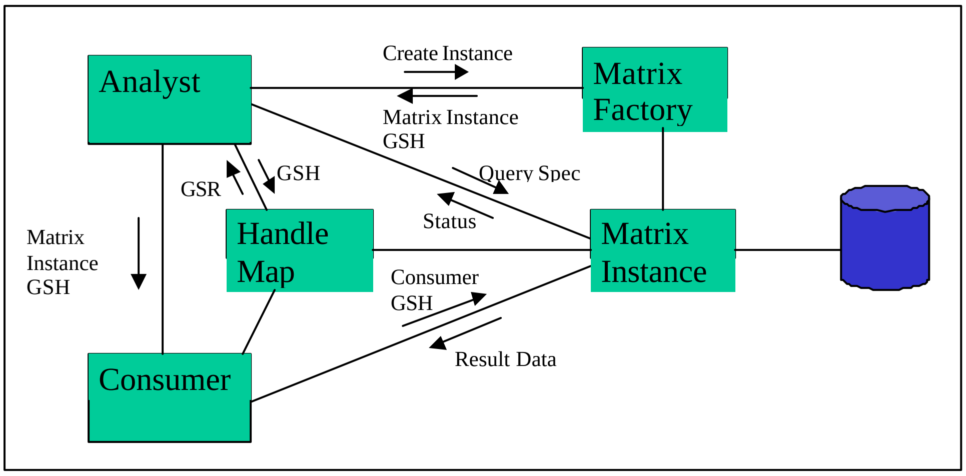

Figure 11 – OGSA Stateful

With the OGSA stateful approach, the infrastructure will maintain the state. The Analyst obtains an instance of the Matrix service (represented by a GSH) and sends a query specification to it. The GSH enables the Consumer to access the instance and the Consumer can send its own GSH to Matrix for delivery of results. There are of course variations on this theme; for example, the Analyst could provide the handle to the Consumer as part of the original query.

It is important to note that the Consumer must have appropriate credentials to make the request on the Matrix service or for the Matrix service to access the Consumer end point.

Issues for handling large data sets

One of the important requirements for a system to access databases on the Grid is the ability to handle large amounts of data efficiently. The data can be in the form of a result set with a small number of rows, each containing a large object (LOB) or a result set with a large number of rows without LOBs or a combination of both. It follows from this requirement that unnecessary copies of the data set should be avoided.

We have identified query, the possible transformation of the result and delivery of the resulting data as key services that a Grid data service should support. These three independent functions have been separated into three independent services.

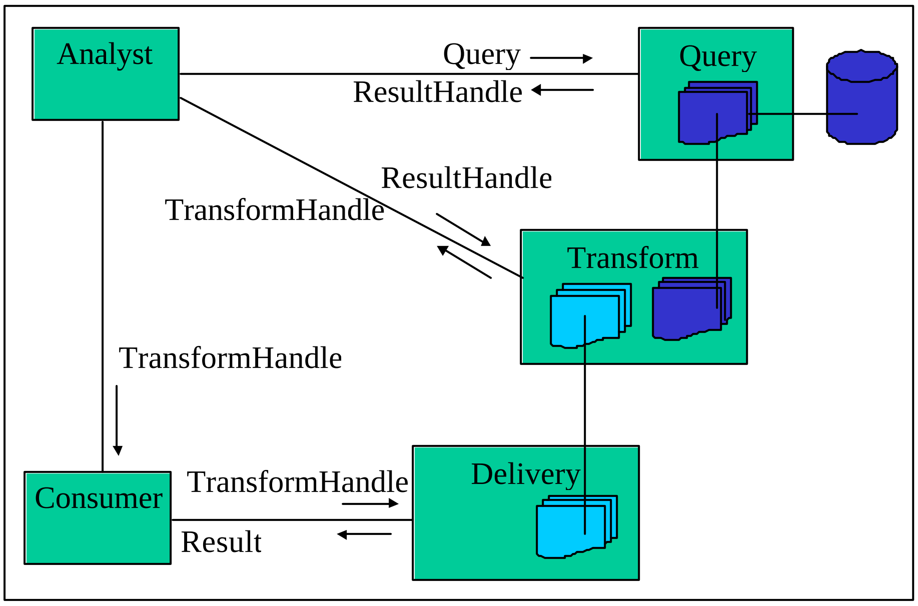

Figure 12 – Handling Data Sets With Distinct Service Instances

However this introduces a possible conflict with the requirement not to copy data unnecessarily if the three services are implemented on separate machines. Once a query is executed against the database the resulting data set has to be cached by the query service in order to present a single state image of the database to the requester. This in itself presents a scalability issue, as similar requests will potentially duplicate data in the cache. Since the transformation service runs on a different host it has to copy the data across first before it can attempt a transform into the required format. Similarly the delivery service has to copy the transformed data across before sending this to the final destination.

As this would represent inefficient use of resources we decided to investigate a way to combine the three services into a single Matrix service but still be flexible about how to compose the three functions.

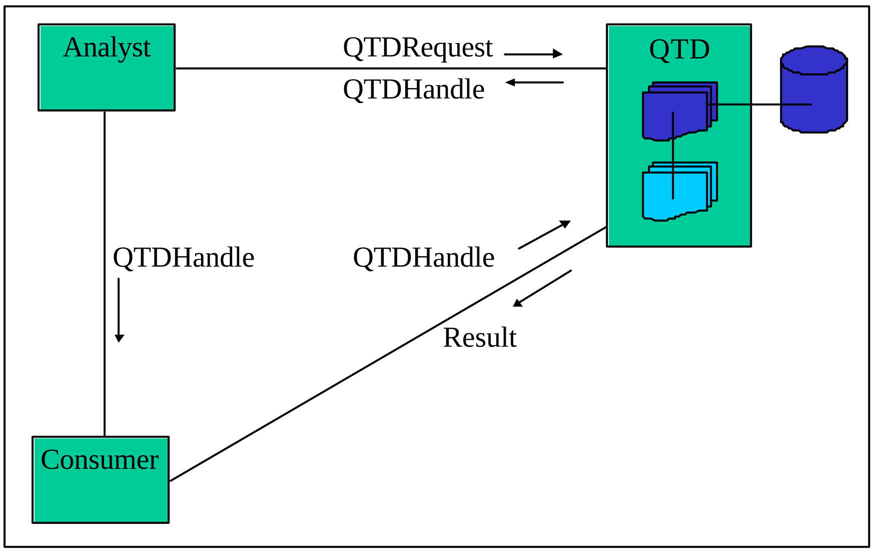

Figure 13 – Consolidated Approach

This is a modular approach that allows new delivery mechanisms or new transformations to be plugged in easily. For each request the user specifies how the building blocks get assembled.

Large Objects

The alternative approaches to storing large objects (LOBs) are as follows:

- The metadata, in this case data about the LOB, is held in a table and the LOB is also held in the same logical table. This approach allows the database management system to control referential integrity, however, though this approach is architecturally sound it can introduce performance difficulties.

- The metadata is held in a table and an external reference is held in the same logical table. The external reference is a pointer to the LOB, which is held outside the relational database for example in a flat file. This approach does not allow for the database management system to control referential integrity.

- The metadata is held in a table and an external reference is held in the same logical table. The external reference is a pointer to the LOB held in a flat file that can optionally be managed as part of the relational database. In this approach the external reference is a URL in a table column with type of DATALINK. The DATALINK type is specified in SQL/MED [41].

It is our assessment that we will have to support all three of the above approaches.

Synchronous and Asynchronous

There are parts of the Matrix infrastructure that will operate synchronously, i.e. execute following a request by the Analyst returning a result with the call response. There are other parts that must operate asynchronously, to support (for example) long-running query or delivery operations.

As soon as the mode of operation moves outside of the simple returning of results with the call response, asynchronous operation is needed.

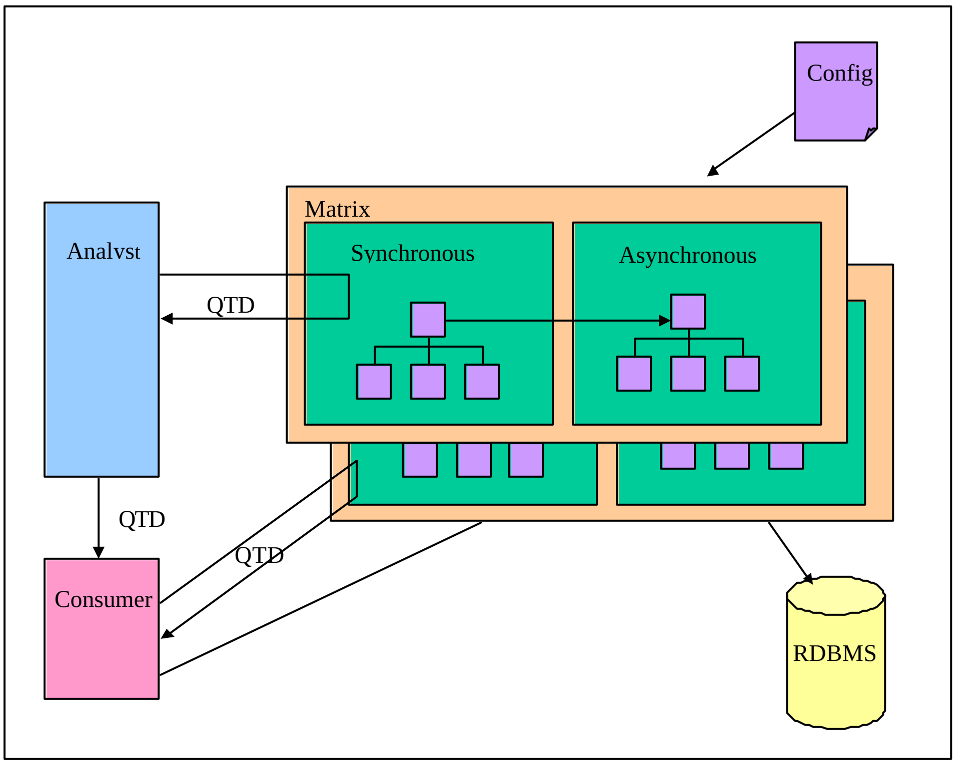

Figure 14 – Synchronous and Asynchronous Operation

Asynchronous operation can be supported through the initiation of control threads either in the called service instance or in a new service instance. In both cases the thread of control undertakes a series of operations in accordance with process defined by the QTD specification in the same way as the synchronous side of the service.

Interface Style

We have discussed interface issues in general without actually defining the interface for the Matrix service. This is intentional, because the external frameworks on which the interface definition depends are currently under development.

The function provided by this interface is clearer and it is interesting to consider how it will be exposed. One alternative is to take an RPC approach and present to the user port types that take many separate parameters. Another is to build a specification, in the form of an XML document, and pass this into the service through a simple port type.

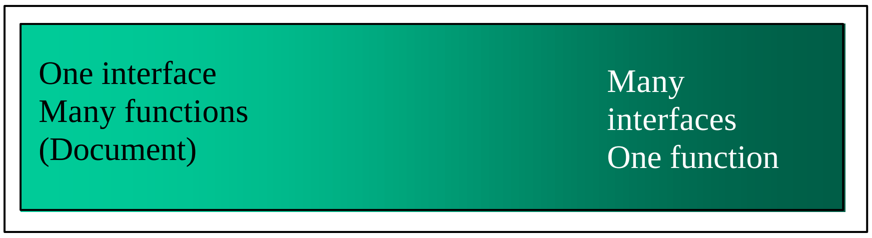

Figure 15 – Interface Styles

Our preference is for a document-style interface for a number of reasons:

The document approach provides a high level of flexibility of function. Specifications encoded as documents can be archived, audited, reused and shared. A document approach more easily allows for the combination of query specifications and metadata along with results.

This reasoning does not of course preclude an RPC interface that wraps a document interface.

Implementation

Chain of Responsibility

As previously discussed, the Matrix functions map onto sub-services such as Query, Transform and Delivery in such a way that flexibility and extensibility are maximized. The 'Chain of Responsibility' design pattern (figure 16) addresses this issue.

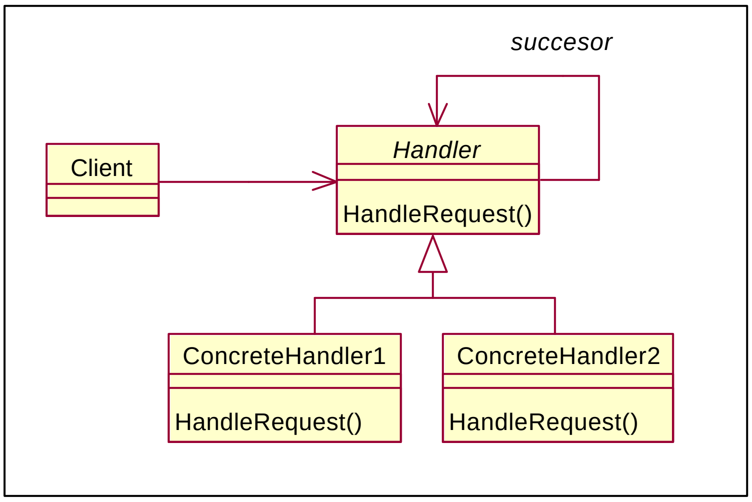

Figure 16 – Chain Of Responsibility Pattern from Gamma et al [42]

This 'Chain of Responsibility' pattern maps directly onto the 'message path' architecture provided by Axis (the Web Services engine under development by Apache **Error! Reference source not found.**).

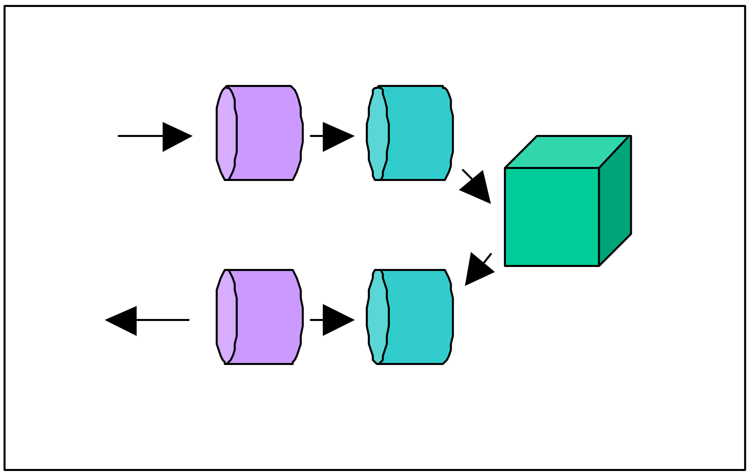

Figure 17 – Axis Message Path

Each of the Query, Delivery and Transport functions maps onto an abstract *Handler* object in this pattern. A Matrix Service can be customised by extending one of the existing handlers or by creating a new handler.

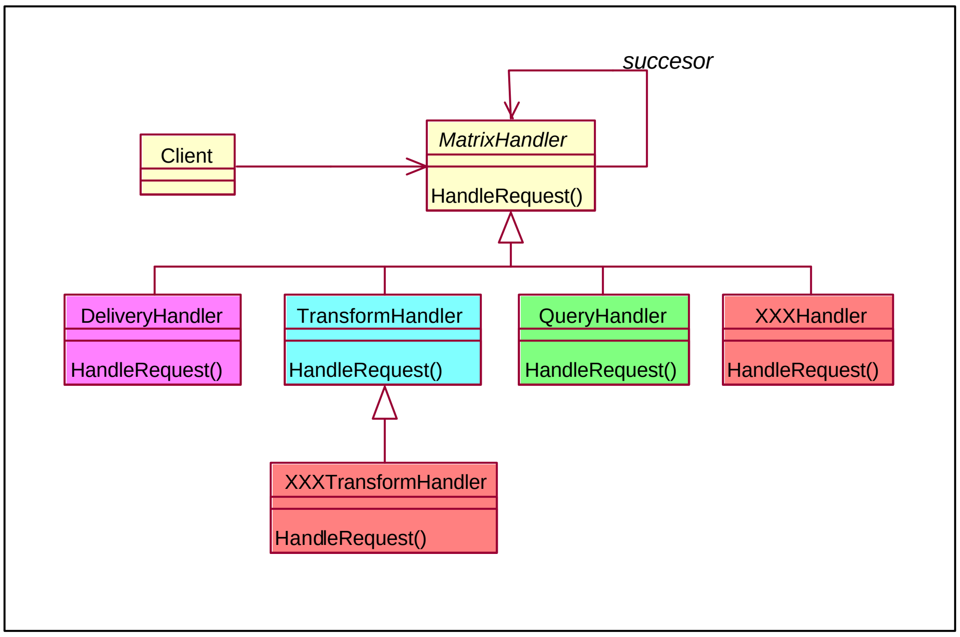

Figure 18 – Handlers Supporting Matrix Functions

QTD Specification

The QTD document is a representation in XML of the specifications required to perform the Query, Transformation and Delivery of data within the Matrix system. An Action Scheduler process is responsible for parsing the QTD document and invoking the actions needed to fulfil the QTD specification. Within the QTD document it is possible to specify a number of operations that must be done, for example a Query of a database, a set of transformations on the results returned and then delivery of the data to one or a number of endpoints.

The major advantage of this approach is the flexibility that the document interface gives. The document interface allows applications to be tolerant of any future changes to the QTD schema should they arise. It is also straightforward to provide RPC style interfaces that could wrap a number of different QTD specifications. This would provide a controlled way to expose the services provided by Matrix.

QTD Structure

The QTD schema is not defined at present; the example presented here is designed to show what a QTD document might look like. This example shows a QTD tag with a processName attribute. For the specified process, it is possible to list a set of activities that must be performed to fulfil the requirements of the QTD document. In the outline document below, three activities are listed and each one would detail the parameters specific to that activity.

```
<QTD processName=“BlueBanana" >
```

```
      <Activity name="SubmitQuery">
      …
      </Activity>

      <Activity name="TransformData">
      …
      </Activity>

      <Activity name="DeliverData">
      …
      </Activity>

</QTD>
```

QTD Example

The delivery service will transport data based on Quality of Service (QoS) criteria. The QoS describes performance metrics and attributes of the transport that must be adhered to. For example, it might be requested that data is transported by the least cost route, but must also be encrypted using a specific type and strength of algorithm.

This QTD specification has four activities to perform:

1. A database query – A simple select statement.
2. Conversion of the returned data via a style sheet transformation – The ProcessStyleSheet activity should convert the data to SVG format.
3. Compression of the SVG format data – Uses gzip to compress the data.
4. Delivery of the data – Data should be delivered to the specified location whilst meeting the QoS criteria for delivery.

```
<QTD processName="BlueBanana">
    <Activity name="SubmitQuery">
        <Param name="QueryString">SELECT myimages.id, myimages.image1,
myimages.image2 from myimages</Param>
    </Activity>

    <Activity name="ProcessStyleSheet">
        <Param name="StyleSheet">ConvertToSVG</Param>
    </Activity>

    <Activity name="Compress">
        <Param name="CompressionType">gzip</Param>
    </Activity>

    <Activity name="DeliverData">
        <Param name="DestinationType">file</Param>
        <Param name="DestinationPath">/home/neil/</Param>
        <Param name="DestinationName">images.xml.zip</Param>
    <Param name="DestinationAddress">serpent.hursley.ibm.com</Param>

        <Param name="QoSDeliveryCost">least cost</QTD:Param>
        <Param name="QoSDeliverySpeed">don't care</QTD:Param>
    </Activity>
</QTD>
```

QTD Interface Styles

The QTD document can be used as the basis of differing styles of interface that range from an RPC style at one extreme to a strongly typed document interface at the other.

Examples might look like:

Strongly Typed Document Style

```
<QTD:QTD processName=“BlueBanana“  xmlns:QTD="http://www.ibm.com" …>
 <QTD:SubmitQuery>
    <QTD:QueryString>SELECT myimages.id, myimages.image1,
myimages.image2 from myimages</QTD:Param>
    <QTD:QueryLanguage>SQL92 <QTD:QueryLanguage>
</QTD:SubmitQuery>
  …
```

Flexible Document Style

```
<QTD:QTD processName=“BlueBanana"  xmlns:QTD="http://www.ibm.com" … >
 <QTD:Activity name="SubmitQuery">
    <QTD:Param name="QueryString">SELECT myimages.id,
myimages.image1, myimages.image2 from myimages</QTD:Param>
    <QTD:Param name="QueryLanguage">SQL92</QTD:Param>
</QTD:Activity>
  …
```

RPC Style

```
<message name=“QueryInputMessage">
    <part name=“queryLanguage" element=“xsi:string"/>
    <part name=“queryLanguage" element=“xsi:string"/>
</message>
…
<portType name=“MatrixPortType">
  <operation name=“query">
    <input message="tns: QueryInputMessage "/>
…
```

Conclusion

The key benefit for any organization participating in open source projects is development of skills and experience within the organization that will be of future value. We have chosen a Web Services architecture and given a good deal of thought to a smooth migration path through to OGSA. In thinking about the interface definitions for Matrix Services we have kept flexibility and extensibility in mind and there is scope for providing interfaces that are essentially independent of the data source.

Our aim is to identify a solution that can be implemented in a series of iterations, to provide a straightforward implementation in the first instance, but one that can be developed through successive iterations into a middleware system that contributes towards the realization of the Grid.

Acknowledgements

We would like to acknowledge the assistance of the UK National eScience Initiative and its Database and Architecture Task Forces; the members of the OGSA-DAI Project; the MyGrid Team; the AstroGrid Team; and the European Data Grid Team.

We would also like to thank Vijay Dialani (University of Southampton); Peter Kunszt (CERN); and Inderpal Narang, Vijayshankar Raman, Susan Malaika, Dan Wolfson,

Rodolphe Michel, Jeff Nick, Jeff Frey, Steve Graham, and Tony Storey (IBM) who assisted us by providing additional input, in workshops, or in reviewing earlier versions of this document.

References

[1] NeSC, IBM, Oracle, eSNW, NEReSC, *OGSA – Data Access and Integration Services*, Project Proposal V 1.1, March 2002, private communication.

[2] I. Foster, C. Kesselman, and S Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organisations*, Int. J. High Perform. Comput. Appl., Vol. 15, No. 3, 2001, http://www.globus.org/research/papers/anatomy.pdf

[3] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Globus Project Technical Report, January 2002, http://www.globus.org/research/papers/ogsa.pdf,

[4] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman, *Grid Service Specification*, Globus Project Technical Report, February 2002, http://www.globus.org/research/papers/gsspec.pdf

[5] T. Hey, and A. E. Trefethen, *The UK eScience Core Programme and the Grid*, 2001

[6] T. Hey, *Towards an eScience Grid Roadmap: The Open Grid Services Architecture*, April 2002

[7] M. Atkinson, et al, *UK Role in Open Grid Services Architecture: UK eScience Core Programme*, Technical Report by UK eScience ATF, March 2002

[8] P. Watson, *Databases and the Grid*, Technical Report CS-TR-755, University of Newcastle, 2001, http://www.cs.man.ac.uk/grid-db/papers/dbg.pdf

[9] D. Pearson, *Data Requirements for the Grid Scoping Study Report*, Technical Report for *Databases and the Grid BOF at* GGF4, Toronto, February 2002, http://www.cs.man.ac.uk/grid-db/papers/Requirements.pdf

[10] N.W. Paton, M.P. Atkinson, V. Dialani, D. Pearson, T. Storey, and P. Watson, *Database Access and Integration Services on the Grid*, Technical Report for *Databases and the Grid BOF at GGF4*, Toronto, February 2002, http://www.cs.man.ac.uk/grid-db/papers/dbtf.pdf

[11] M.P. Atkinson, *Comments and Additions to A Model for Database Access and Integration on the Grid*, Technical Report on first draft of [10], January 2002.

[12] M.N. Alpdemir, N.W. Paton, and A.A.A. Fernandes, *An Investigation on the Integration of the Globus Toolkit with Relational DBMSs*, Technical Report, University of Manchester, March 2002, private communication.

[13] D. Gannon, K. Chiu, M Govindaraju, and A Slominski, *An Analysis of the Open Grid Services Architecture*, Technical Report commissioned by the UK eScience Core Programme, Indiana University, 2002, http://www.extreme.indiana.edu/~gannon/OGSAanalysis3.pdf

[14] P. Z. Kunszt, *The Open Grid Services Architecture: A Summary and Evaluation*, Technical Report commissioned by the UK eScience Core Program, CERN, April 2002,

http://umbriel.dcs.gla.ac.uk/NeSC/general/teams/OGSAreviewPeterKunszt.pdf

[15] I. Narang and V. Raman, *Database Issues in Grid Computing: IBM Research Perspective*, Technical Report *for OGSA-DAI Workshop*, Edinburgh, April 2002

[16] *JDBC 2.1 Specification*, Sun Microsystems, October 1999, ftp://ftp.javasoft.com/pub/jdbc/9128374/jdbc2_1-spec.pdf

[17] J. Ellis, L. Ho, and M. Fisher, *JDBC 3.0 Specification*, Sun Microsystems, October 2001, ftp://ftp.javasoft.com/pub/spec/jdbc/d0o9FR30dsofe/jdbc-3_0-fr-spec.pdf

[18] C. Fan, J. Funderburk, H. Lam, J. Kierman, E. Shekita, and j Shanmugasundaram, *XPERANTO: Bridging Relational Technology and XML*, February 2002

[19] H. Katz, *An Introduction to Xquery*, February 2002, http://www.ibm.com/developerworks/library/x-xquery.html

[20] M. Papiani, J. L. Wason, A. N. Dunlop, and D. A. Nicole, *A Distributed Scientific Data Archive Using the Web, XML and SQL/MED*, ACM SIGMOD Vol. 28, No. 3, September 1999, http://www.acm.org/sigs/sigmod/record/issues/9909/papiani.pdf

[21] *MyGrid Proposal Document* - http://mygrid.man.ac.uk/more.html

[22] *Astrogrid Usecase list* - http://wiki.astrogrid.org/bin/view/VO/UseCaseList

[23] C. Page, *Database Technology For Astrogrid: A Discussion Document*, 09 May 2002 - http://www.star.le.ac.uk/~cgp/ag/dbstatus.html

[24] C. Page, *Indexing the Sky*, 09 May 2002 - http://www.star.le.ac.uk/~cgp/ag/skyindex.html

[25] R. Williams, et al, *VOTable: A Proposed XML Format For Astronomical Tables*, Version 1.0, 15 April 2002 - http://vizier.u-strasbg.fr/doc/VOTable/

[26] J. Gray, D. Slutz, A. Szalay, A. Thakar, J. van den Berg, P. Kunszt, and C. Stoughton, *Data Mining the SDSS SkyServer Database*, Microsoft Technical Report, MSR-TR-2002-01, January 2001, http://research.microsoft.com/~Egray/Papers/MSR_TR_O2_01_20_queries.pdf

[27] *Java Developer Connection, Early Access, JDBC Rowset* - http://developer.java.sun.com/developer/earlyAccess/crs/index.html

[28] N. C. Hong, *Writing GSI enabled Web Services* - http://www.epcc.ed.ac.uk/~neilc/gsiws/

[29] Oracle Application Developers Guide, *XSQL Pages Publishing Framework* - http://download-uk.oracle.com/otndoc/oracle9i/901_doc/appdev.901/a88894/adx10xsq.htm

[30] ISO-ANSI Working Draft - *XML-Related Specifications (SQL/XML)*, Mar2002

[31] J. Melton, J-E. Michels, V. Josifovski, K. Kulkarni, P. Schwartz, K. Zeidenstein, *SQL and Management of External Data*

[32] M. Atkinson, V. Dialani, L.Guy, I. Narang, N. Paton, D. Pearson, T. Storey, and P. Watson, Grid Database Access and Integration Requirements and Functionalities (Draft), 04 July 2002, http://www.cs.man.ac.uk/grid-db/papers/dairf.pdf

[33] A. Krause, K. Smyllie, R. Baxter, *Grid Data Services Specification for XML Databases Version 1.0*, 19 June 2002, http://www.cs.man.ac.uk/grid-db/papers/GXDSspec.pdf

[34] J. Smith, A. Gounaris, P Watson, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou, *Distributed Query Processing on the Grid*, http://www.cs.man.ac.uk/grid-db/papers/dqp.pdf

[35] V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, and C. Baru, *Data Access and Management Services on Grid*

[36] European DataGrid Work Package 2 {Project Spitfire), http://hep-proj-spitfire.web.cern.ch/hep-proj-spitfire/server/doc/index.html

[37] Global Grid Forum, http://www.gridforum.org/

[38] IBM - OGSA - DAI – WP6 – Scenarios and Patterns, to be published

[39] Grid Security Infrastructure, http://www.globus.org/security/

[40] Web Services Security, http://www-106.ibm.com/developerworks/library/ws-secure/

[41] DATALINK, http://www.almaden.ibm.com/cs/datalinks/

[42] E Gamma, R Helm, R Johnson and J Vlissides, Design Patterns: elements of reusable object-oriented software, Addison-Wesley, Reading, Mass, 1994

[43] Apache Axis (SOAP Implementation), http://xml.apache.org/axis/

[44] Apache Xalan (XSL Transformation engine), http://xml.apache.org/xalan-j/index.html

[45] Secure Sockets Layer 3.0 specification, http://wp.netscape.com/eng/ssl3/

[46] Web Services Security (WS-Security), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp

[47] M. Atkinson, *OGSA-DAI Project Work Package 2 – A Strategy for Architectural Development and an Outline Architecture*, Version 0.2, 28-31 May 2002, private communication.