Transaction-Based Grid Database Replication

Yin Chen[1], Dave Berry[1], Patrick Dantressangle[2]

[1]National e-Science Centre, Edinburgh, UK
[2]IBM, Hursley Lab, Winchester, UK

Abstract

We present a framework for grid database replication. Data replication is one of the most useful strategies to achieve high levels of availability and fault tolerance as well as minimal access time in grids. It is commonly demanded by many grid applications. However, most existing grid replication systems only deal with read-only files. By contrast, several relational database vendors provide tools that offer transaction-based replication, but the capabilities of these products are insufficient to address grid issues. They lack scalability and cannot cope with the heterogeneous nature of grid resources.

Our approach uses existing grid mechanisms to provide a metadata registry and to make initial replicas of data resources. We then define high-level APIs for managing transaction-based replication. These APIs can be mapped to a variety of relational database replication mechanisms allowing us to use existing vendor-specific solutions. The next stage in the project will use OGSA-DAI to manage replication across multiple domains. In this way, our framework can support transaction-based database synchronisation that maintains consistency in a data-intensive, large-scale distributed, disparate networking environment.

1. Introduction

A replication system maintains multiple copies of data objects, synchronising any changes to that data. Replication has been widely employed for enhancing system accessibility (e.g. recovering data from disaster), data consolidation (for central audit or analysis), data distribution (for balancing access load, or for offline query), etc. In the grid community, replication has attracted a great deal of interest [12, 13, 14]. Since many applications, e.g. high-energy physics, biology, and astronomy, often generate large amounts of data and the users who access this data are usually physically distributed, replicating data locally can significantly reduce the data access latency, as well as achieve high levels of availability and fault tolerance.

Outside the grid community, several relational database vendors provide replication tools. Sybase replication [1] was among the early solutions, emerging in 1993. Since then, most database companies, Oracle [2], Sybase, DB2 [3, 4], MySQL [5], and MS SQL Server [6], offer a variety of products ranging from simple to complicated replication mechanisms, including complete or partial copying, synchronous or asynchronous copying, etc. However, new difficulties come with the challenges of grids. The capabilities of relational database replication products appear to have two significant limitations when used in a grid environment. Firstly, they don't address the scalability issue. For example, DB2 SQL replication uses DRDA (Distributed Relational Database Architecture) to deliver data, which cannot cope with transferring large datasets [7]. Secondly, they assume that the network environment is homogenous; they don't handle moving data among heterogeneous databases in virtual organisations.

Several Grid-based data replication systems exist, among them the EGEE Data management Service [8], the Globus Data Replication Service (DRS) [9], the Lightweight Data Replicator (LDR) [25], and the Storage resource Broker (SRB) [10]. These tools support transferring large datasets across the network, offering more scalability. For example, LDR, designed and developed for the Laser Interferometer Gravitational Wave Observatory (LIGO), is capable of coping with replication task of moving over 50 terabytes of data across ten locations [25]. Nevertheless, most of these tools only deal with (read-only) files rather than transaction-based systems. SRB provides simple operations to manage data movement among heterogeneous databases, yet it doesn't support transaction-based replication.

Our review of the literature reveals that no prior effort, to our knowledge, has been made to build a transaction-based grid database replication model. So, the primary motivation for this work is to make a first attempt at a practical implementation.

The challenges include but not limited to: how to scale up the database replication to support access across multiple domains; how to

move large data objects; how to propagate data changes among different database systems; and how to support all kinds of user requirements (different users have different reasons to use replication).

In this paper, we present our framework for transaction-based grid database replication. In this framework, a metadata registration mechanism is employed to enable the visibility of existing data sources and their replicas. Grid transfer tools are used to make the initial copy of the data when creating a replica. A unified API is provided to interact with a number of relational database replication mechanisms to propagate the changes of data, and maintain consistency between the sources and replicas.

In this way, we are able to integrate the relational database replication model and the grid replication model. This brings us two significant benefits. Firstly, it helps traditional database users to move their legacy software onto grids smoothly, thus allowing them to address more challenges and maximise their computational benefits at minimal cost. Secondly, it extends the range of functionality available to grid users, allowing them to use more sophisticated data management systems, including modifiable data, without losing the benefits of grid-based replication.

The rest of article is organised as follows. Section 2 describes a specific grid scenario which we target when designing the grid database replication model. Section 3 compares two existing replication models. Section 4 describes our system design in detail. Section 5 discusses the implementation issues. We end the paper with conclusions in Section 6.

2. Case Study

SIMDAT (www.scai.fraunhofer.de/simdat.html) is a European Commission Sixth Framework Programme project producing data grids for process and product development using numerical simulation and knowledge discovery. The design of SIMDAT is based on a series of interconnected nodes. The demonstration system supports the discovery and retrieval of datasets from following sites: ERA40 data from ECMWR, climate time series from Deutscher Wetterdienst, model output from Météo-France, aviation weather reports from the UK met Office, and satellite images from EUMETSAT. Each site comprises a local database (storing data), a portal (for accessing data held in one or more data repositories and offering cataloguing services), and a catalogue (storing metadata). The catalogues held at the various nodes need to be fully synchronised and any data available on any of the nodes can be discovered and retrieved from any of the web portals. [11]

SIMDAT provides us a typical use scenario for grid database replication. It requires efficient and reliable replication software to synchronise the metadata catalogues in 5 distributed organisations. Similar requirements can be found in many other grid applications. With the demand continually increasing, there is a great need to build an effective and widely applicable implementation.

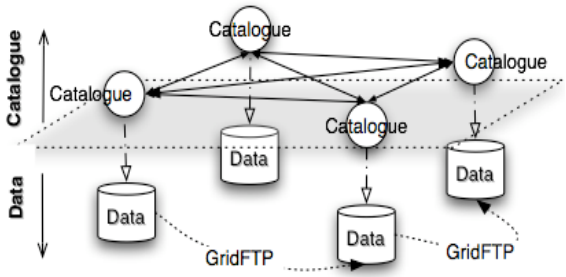

Fig 2.1 Architecture of SIMDAT

3. Existing Replication Models

3.1 Relational Database Replication Model

Most relational database replication software, (e.g. Sybase replication [1], Oracle Advanced Replication [2], DB2 SQL replication and Q-replication [3, 4], and MS SQL Server replication [6]) adopt a 'publish-and-subscribe' structure (illustrated by Fig 3.1). In this model, a change-capture component detects data changes in the source database(s), usually through a trigger or log mechanism, and either immediately or periodically publishes them into temporary tables. The temporary tables can be created in the source database, or the target database, or in a separate database server. A change-apply component then reads contents from the temporary tables and applies them to target database(s) in a predefined schedule.

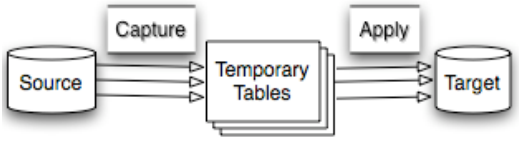

Fig 3.1 Relational Database Replication Model

The replication processes are managed by a replication server. The replication server can communicate with other replication servers, hence forming a peer-to-peer network which is capable of supporting a range of replication use

cases, such as warm-up standby replication (for disaster recovery), Mission Critical Applications (which are those where failure of execution or faulty execution may have catastrophic results), and Mass Deployment (in which applications distribute database infrastructure, data, and front-end application to a large number of users), etc.

However, the implementations of relational database replication are insufficient to address grid issues. In most replication scenarios, there is a need to make an initial copy of the whole dataset before applying data changes. In the grid, datasets to be replicated are often very large, and most relational database replication tools are unable to cope. Another drawback of this model is that there is no resource discovery mechanism and it is difficult to optimise replication strategies. Many grids also cross organisational boundaries and this introduces the need for security mechanisms that operate in a highly distributed, heterogeneous dynamic network environment.

3.2 The Grid Replication Model

Many existing grid replication systems, e.g. Globus (DRS), SRB and the EGEE Data Management Service, employ a 'pull' method to move data. As shown as Fig 3.2, in this model, a metadata catalogue keeps an index of source files and the mapping to their physical locations. (Grid replication systems differ in the exact design of this catalogue: some split it into separate components for managing metadata and managing replica locations; some use a central catalogue while others use a distributed catalogue. These differences are not significant to the design of the replication mechanism itself.) And GridFTP is used to deliver the datasets. When requested, the replication service will search the metadata catalogue to locate the file(s), and then start the GridFTP service to fetch the datasets locally.

Data change-capture and change-apply elements are absent from this model. It does not support writable data and hence does not have to maintain the consistency between source data and replicas.

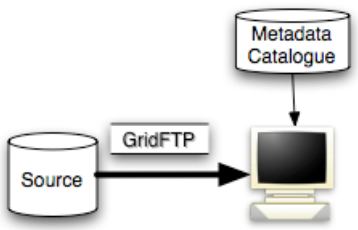

Fig 3.2 Existing Grid Replication Model

4. Our Approach

4.1 The Grid Database Replication Model

After analysing the requirements of our scenario and the technical restrictions, we propose the Grid Database Replication Model, as depicted in Fig 4.1.

The replication actions are managed by the **Replication Control Service,** which comprises a set of grid services. A **Metadata Registry** stores meta-information of data resources and their replicas, providing a means of data resource discovery. The **Transfer Service** engages a powerful grid transfer mechanism, e.g. GridFTP, to make the initial copying of the snapshot of the source. A number of **relational database replication mechanisms,** e.g. DB2 replication, Oracle replication, MS SQL Server replication etc, can be integrated in the model as plug-ins to synchronise data updating and monitor the replication processes. This allows users to select a preferred replication mechanisms depending on concrete operational conditions and their replication requirements.

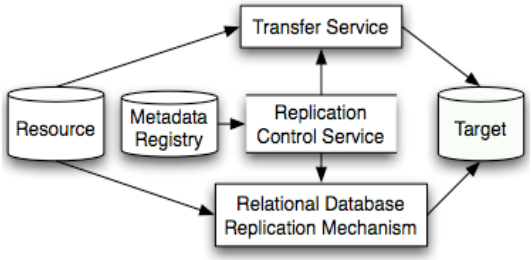

Fig 4.1 The Grid Database Replication Model

The design details of the **Replication Service** are illustrated in Fig 4.2. There are five principal components:

- the **Searcher** is a semantic metadata search engine, which queries the *Metadata Registry* based on both content and performance requests;
- the **Selector** uses an optimisation model to return the best choice of replica based on a suitable cost function;
- the **Registry Manager** records the information of data sources and new replicas in the metadata registry;
- the **Initiator** is an interface to communicate with the third-party services such as the *Transfer Service*;
- the **Starter** implements a group of APIs for interacting with the relational database replication mechanisms.

The control workflow is described as follows. The client requests are captured by the

Searcher, which performs a semantic query on the metadata registry (see step 1). The output of metadata searching will be a list of all possible resources. This will be delivered as the inputs to the *Selector*, which returns the most applicable replication resource (see step 2). (Some existing works such as [21, 22, 23] investigate possible cost functions. We will evaluate the algorithms and apply a suitable one.) Then, the control process will reach the *Registry Manager*. The *Registry Manager* updates the metadata registry with the information of the new replica (see steps 3 and 4). Meanwhile, the *Initiator* will be triggered, which communicates the user chosen *Replication Database Replication Mechanism*, e.g. DB2 SQL replication, to configure the source and replica databases (see steps 5 and 6). When the replication environment has been prepared, the *Initiator* will start the *Transfer Service*, which implements a powerful data transferring tool, like GridFTP, to make the initial copy of the source data on the replica database (see steps 7, 8 and 9). Afterwards, the *Initiator* will notify the *Starter* to start and pass control to the indicated relational database replication, which will continually monitor the source database and apply data updating to the replica database (see steps 10, 11, 12, 13).

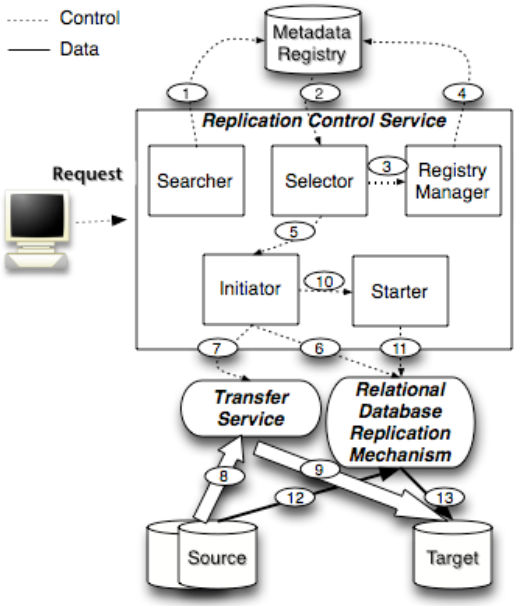

Fig 4.2 The Replication Control Service Architecture and the Control Workflow

4.2 Metadata Registry Mechanism

The Metadata Registry is deployed as an embedded database and wrapped by a grid middleware service, e.g. OGSA-DAI [16], which enables it to be exposed onto the grid. The metadata used in the system includes content and configuration information of the replication resources such as:

- Domain name
- Transfer protocols supported (e.g. GridFTP)
- Replication protocols supported (e.g. Oracle replication, DB2 replication, file copy, etc.),
- Names of database(s) to be replicated,
- Names of table(s) to be replicated,
- Schema information for the relevant data

Further metadata information includes Quality of Service information for the data resource, including, responsiveness (e.g. response time), throughput, reliability (measured by the percentage of time a service can correctly execute), availability (measured by the fraction of the time a resource is available for use), and security constraints (security level, degree of trust, etc.).

4.3 Interaction with the Transfer Service

The **Initiator** of the Replication Control Service interacts with third-party services to transfer data. In our prototype implementation we are using GridFTP to deliver large blocks of data, particularly the initial copy of a replica. GridFTP supports parallel and striped data transfer and partial file access, presenting significant improvement in transfer time, network throughput and utilisation [15]. Work is ongoing in the Open Grid Forum to standardise a control interface to grid transfer protocols such as GridFTP, BBFTP and similar systems.

GridFTP is a file transfer protocol and cannot be directly applied to transactions. We solved this problem by utilising the data dump tool offered by most of the relational database products. The source data are firstly exported as files, then shipped to the target site through multithreaded GridFTP transfers, and finally loaded into the replica database.

4.4 Integrated Relational Database Replication

The **Starter** of the Replication Control Service implements a group of high-level APIs which abstract the common behaviours of relational database replication. This group of APIs provides a unified interface and many popular database replication mechanisms can be plugged into it. After studying functionalities and operations of several popular relational database replication systems, including DB2 replication, Sybase replication, Oracle advanced

replication, and MySQL replication, we have defined the following key classes:

- *Connection*: a set of classes used for making connection or disconnecting with a remote server.
  `ConnectRemoteHost(),`
  `DisconnectRemoteHost(),`
- *Replica Creation*: a set of classes used for creating or removing a new database for replication.
  `CreateReplicaDatabase(),`
  `DropReplicaDatabase(),`
- *Configuration*: a set of classes used for setting up replication environment or clean up the configurations.
  `ConfigReplication()`
  `CleanUp()`
- *Starting and Stopping Replication*: a set of classes used for starting and stopping a replication process.
  `StartReplication()`
  `StopReplication()`
- *Monitoring Replication Processes*: a set of classes used for observing the status and performance of a replication process.
  `MonitorReplication()`

These APIs unify the existing relational database replication systems and hide the underlying complexity of the different data access mechanisms. They include functionality for creating replicas and for monitoring traffic between them. The configuration API includes facilities for specifying the degree of consistency to be maintained between replicas. For example, if a record is updated in one replica and then read from another, the configuration settings will specify how up-to-date the data read will be in the worst case.

4.5 Cross-Domain Replication

An important feature of the Grid Database Model is that it also supports cross-domain database replication.

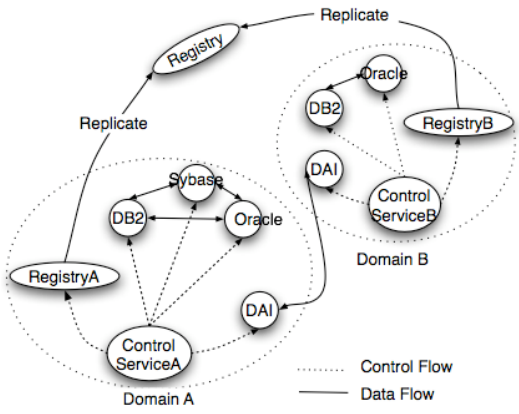

Fig 4.3 Cross-Domain Replication

There are 3 strategies for cross-domain replication resource discovery:

(1) One can directly query a remote metadata registry;
(2) The user can synchronise multiple metadata registries using the developed grid database replication technology, hence forming a peer-to-peer resource discovery overlay; or
(3) As shown in Fig 4.3, each local metadata registry can partially replicate the information to be shared to the public registry, to construct a hierarchical registries discovery mechanism. Partial data replication allows users to only publish the shared information and protect private information from being visible externally, thus minimising security risk as well as maximising collaboration benefits.

Some advanced relational database replication systems, e.g. DB2 Q-replication, provide means to work across organisational boundaries. These can plug in to our model directly.

If the relational database replication software does not have such support, e.g. DB2 Data Propagator, the implementation of our GDR model will have to provide the replication functionality itself. There are several possible approaches.

One simple mechanism would be to use *Transfer Service* to ship the replication data as well as the initial copy. In this case, the Replication Control Service at the source site starts the change-capture process of the specified relational database replication, which casts the source updates into the temporary tables. The *Transfer Service* then reads from the temporary data pool and transports them to the temporary data pool in the replica site. The Replication Control Service in the replica site is invoked to start the change-apply process to read from the temporary tables and apply the changes.

Alternative approaches would be to build replication functionality on top of event notification systems such as WS-Eventing[17], IBM's Message Queue [18], or Info-D [24].

An implementation using any of these approaches would be independent of a particular database replication system. This would support data movement among heterogeneous databases as well as across domains. We plan to use OGSA-DAI as a framework to implement this functionality. OGSA-DAI (www.ogsadai.org.uk) is grid middleware, compliant with two popular web service specifications, WS-I and WSRF, and distributed

with both the Globus Toolkit and the OMII-UK middleware distributions [16].

4.6 Security

A key characteristic of Grids is that they provide authentication and authorisation systems across organisational boundaries. A grid-based replication system must work with the relevant security mechanisms.

We assume that the replicas in our model are deployed within a global system for authenticating users across domains, such as a PKI system based on X.509 digital certificates. This will identify which user is requesting access to a replica, if necessary mapping the global user identifier to a local one.

Authorisation to perform an operation on a resource is a requirement above and beyond simple authentication. In a grid environment, an initial authentication check may be performed by a virtual organisation management system. The replica itself may perform more detailed checks.

It follows that the Replication Control Service must pass authentication and authorisation credentials to the resources that it uses. The Searcher and Selector components should query the virtual organisation management service so that they only return replicas that the user is authorised to access. The system must also allow operations to be rejected by the replica itself and provide a fallback to query other replicas instead.

5. Implementation of GDR

We are progressively implementing each component of the Replication Control Service in a prototype implementation.

We use DB2 SQL Replication as an example to map the relational database replication plug-in interface we described in section 4.4. We have successfully developed a DB2 SQL replication plug-in and through the Starter component we can control the replication from configuration of the replication environment up to starting applying data changes.

The following describes the technical details of manipulating DB2 SQL replication from the Replication Control Service. We interact with the replication processes by calling DB2 ASNCLP [19] and CLP [20] programs. Both are command-line interfaces: ASNCLP is for administration of DB2 SQL replication, and CLP is for database management.

Setting up DB2 SQL Replication environment involves several steps. As shown in Fig 5.1, the key processes in the configuration flow include: creation of Capture control tables if they don't exist; creation of Apply control tables if they don't exist; registration of tables to be replicated; creation of subscription set(s) and members of subscription set(s), which store the bindings of sources and targets.

Input: *Source database and replica database login information, tables to be replication, and the frequency of applying changes.*

Output: *a replication plan document written in ASNCLP and CLP.*

Workflow:

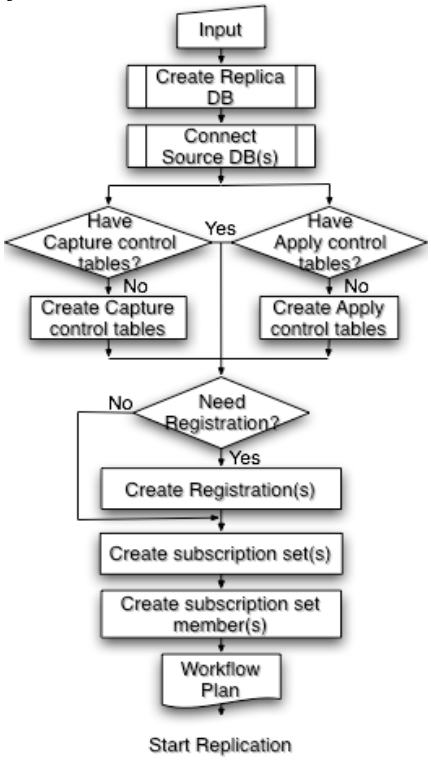

Fig 5.1 Configure DB2 SQL replication

Based on user requirements, we can generate various working plans for different replication strategies and configurations. Fig 5.2 is an example of a configuration and execution plan which is generated during the replication process. In this scenario, a table, named METACATALOGUE, in the source database, SIMDAT1, will be replicated to the replica database, SIMDAT2. The target table's name is NESCRPL_TRGMETACATALOGUE. Any changes to source table METACATALOGUE will be reflected in NESCRPL_TRGMETACATALOGUE within one minute.

```
#Connect Source database (CLP commands)
catalog tcpip node srcnode remote srchst server 50000;
catalog database SIMDAT1 as srcdb at node srcnode;

#Create Replica Database (CLP commands)
catalog tcpip node cpynode remote cpyhst server 50000;
create database SIMDAT2;

#Create Apply control table (ASNCLP commands)
set server control to db SIMDAT2 id usr password “usrpd”;
set run script now stop on sql error on;
create control tables for apply control server in uw others
tsasn100;

#Create Registration (ASNCLP commands)
set server capture to db METACATALOGUE id admin
password “admin”;
set run script now stop on sql error on;
create registration (admin.METACATALOGUE) differntial
refresh stage cdemployee;

#Create subscription set and its member(ASNCLP commands)
set server capture to db SIMDAT1 id admin password “admin”;
set server control to db SIMDAT2 id usr password “usrpd”;
set server target to db SIMDAT2 id usr password “usrpd”;
set run script now stop on sql error on;

create subscription set setname SET00 applyqual AQ00
activate yes timing interval 1 start date “2007-04-16”
time”17:08:00.000000”;

create member in setname SET00 applyqual AQ00 activate
yes source admin.METACATALOGUE target name
usr.NESCRPL_TRGMETACATALOGUE definition in
tstrg00 create using profile TBSPROFILE type;
```

Fig 5.2 A Sample Replication Plan

The working plan is sent to DB2 SQL Replication engines. Both source database and target database replication engines will be involved to execute the plan. They will generate the necessary working tables and a shared data pool, and start change-capture program and change-apply program respectively. The result of executing the sample plan is shown in Fig 5.3. We can see a new replica database, SIMDAT2, has been created, together with a set of staging tables and the replica table, NESCRPL_TRGMETACATALOGUE. The replica table will be refreshed every minute.

6. Future Work

Our current implementation has demonstrated our framework with one database replication system. We intend to extend and evaluate this work in several ways.

The next stage of our work will be to implement a second vendor-specific replication system, such as Oracle. This will test whether our API is generic enough to cope with systems from different vendors.

As discussed in the text, we plan to implement a replication plug-in that uses OGSA-DAI. This will provide a replication capability that functions across domains and with heterogeneous databases.

We also wish to evaluate and enhance algorithms for selecting a replica from those available, based on a combination of resource metadata and information from monitoring systems.

We will apply the developed approach to real use cases, such as the SIMDAT case described above.

Finally, we will develop techniques for measuring performance and scalability. This will require specific development work, since we are unaware of any existing comparable technology.

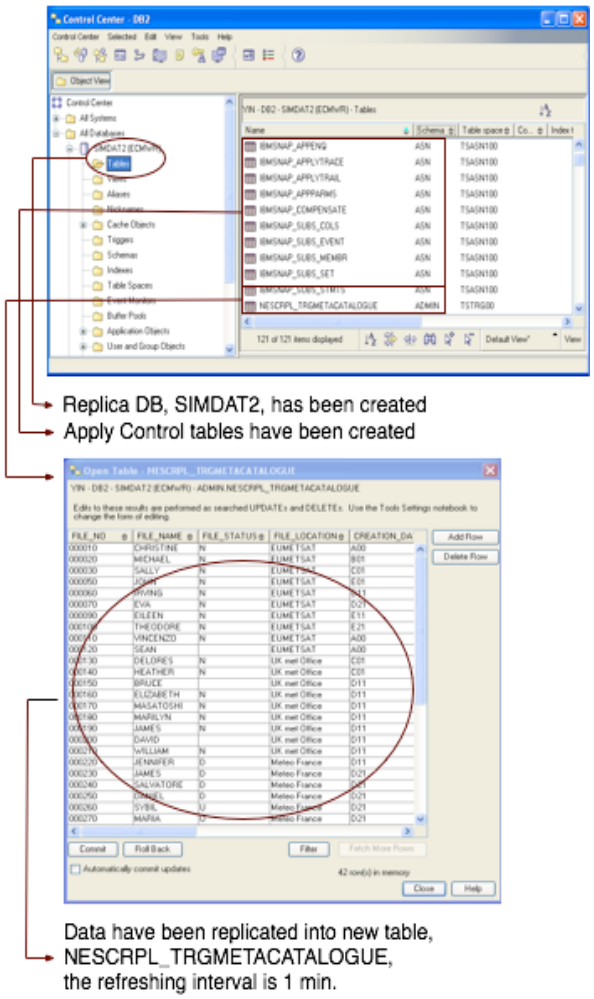

Fig 5.3 The Execution Results

7. Conclusion

This paper discusses a framework for transaction-based grid database replication. In the proposed model, a metadata registry

provides a means for resource discovery; a grid transfer service is used to deal with large amounts of data movement, in particular the initial copy of a replica; and we integrate a range of relational database replication mechanisms to synchronise changing data. Our model will also support grid-based file replication.

We define a group of high-level APIs to abstract the underlying complexity of the different data replication mechanisms. Our framework can interface to existing, vendor-specific, database replication systems or to new mechanisms that support replication in a heterogeneous environment. In this way, our framework can support database replication in a scalable, heterogeneous grid network environment.

References

[1] *Replication Strategies: Data Migration, Distribution and Synchronization,* Sybase White Paper, 2003.

[2] M. Pratt, *Oracle9i Replication*, Oracle White Paper, 2001.

[3] L. Gu, L. Budd, A. Cayci, C. Hendricks, M. Purnell, C. Rigdon, *A Practical Guide to DB2 UDB Data Replication V8*, IBM Redbooks.

[4] A. J. Ciccone & B. Hamel, *The Next Generation: Q Replication*, DB2 Magazine, Quarter 3 2004 Vol. 9, Issue 3.

[5] J. D. Zawodny, D. J. Balling, *High Performance MySQL: Optimization, Backups, Replication & Load Balancing*, O'REILLY, 2004.

[6] R. Sharma, *Microsoft SQL Server 2000: A Guide to Enhancements and new Features*, Addison-Wesley Professional, 2002.

[7] C. Mullins, *DRDA*, dbazine.com, 2006, www.dbazine.com/db2/db2-mfarticles/mullins-drda.

[8] P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, *Data Grid Replica Management with Reptor*, In Proceedings of 5th international conference on parallel processing and applied mathematics, 2003.

[9] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, B. Moe, *Wide Area Data Replication for Scientific Collaborations*, In Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, 2005.

[10] C. Baru, R. Moore, A. Rajasekar, M. Wan, *The SDSC Storage Resource Broker*, In Proceeding of CASCON'98 Conference, Nov.30-Dec.3, 1998, Toronto, Canada.

[11] *D.2.1.2 Consolidated Requirements Report, Roadmap and SIMDAT Infrastructure Design*, SIMDAT Project Report, 2004.

[12] PPDG Project, www.ppdg.net

[13] C. Nicholson, *Dynamic Data Replication in LCG 2008*, UK e-Science All Hands Conference, Nottingham, September 2006.

[14] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stochinger & K. Stockinger, *Data Management in an International Data Grid Project*, 1st IEEE/ACM Int'l. Workshop on Grid Computing (Grid'2000), Bangalore, India, December 2000.

[15] B. Konya, O. Smirnova, *Performance Evaluation of the GridFTP within the NorduGrid project*, 10/1/2001.

[16] M. Antonioletti et al., *The Design and Implementation of Grid Database Services in OGSA-DAI*. Concurrency and Computation: Practice and Experience, Volume 17, Issue 2-4, Pages 357-376, February 2005.

[17] Don Box et al., *Web Service Eventing*, W3C Member Submission, 15/03/2006.

[18] IBM WebSphere MQ, www-306.ibm.com/software/integration/wmq/

[19] *ASNCLP Program Reference for Replication and Event Publishing version 8.2*, DB2 UDB Product Manuals.

[20] J. Forrest, *CLP User Guide*, 2004, IBM Product Manuals.

[21] H. Lamehamedi, B. Szymanski, Z. Shentu & E. Deelman, *Data Replication Strategies in Grid Enviornments*, 2002, In Proceeding of the 5th International Conference on Algorithms and Architecture for Parallel Processing.

[22] M. Cigian, L. Hluchy, *Towards Scalable Grid Replica Optimization Framework*, in Proceeding of the 4th International Symposium on Parallel and Distributed Computing, 2005.

[23] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, F. Zini, *Simulation of Dynamic Grid Replication Strategies in OptorSim*, 2002, In Proceeding of the Third International Workshop on Grid Computing.

[24] Steve Fisher et al., *Information Dissemination in the Grid Environment – Base Specifications*, OGF submission, May 2007.

[25] L. Liming, *Large-Scale Data Replication for LIGO*, ClusterWorld submission, July 2005.