



Universita'  
degli Studi di  
Lecce



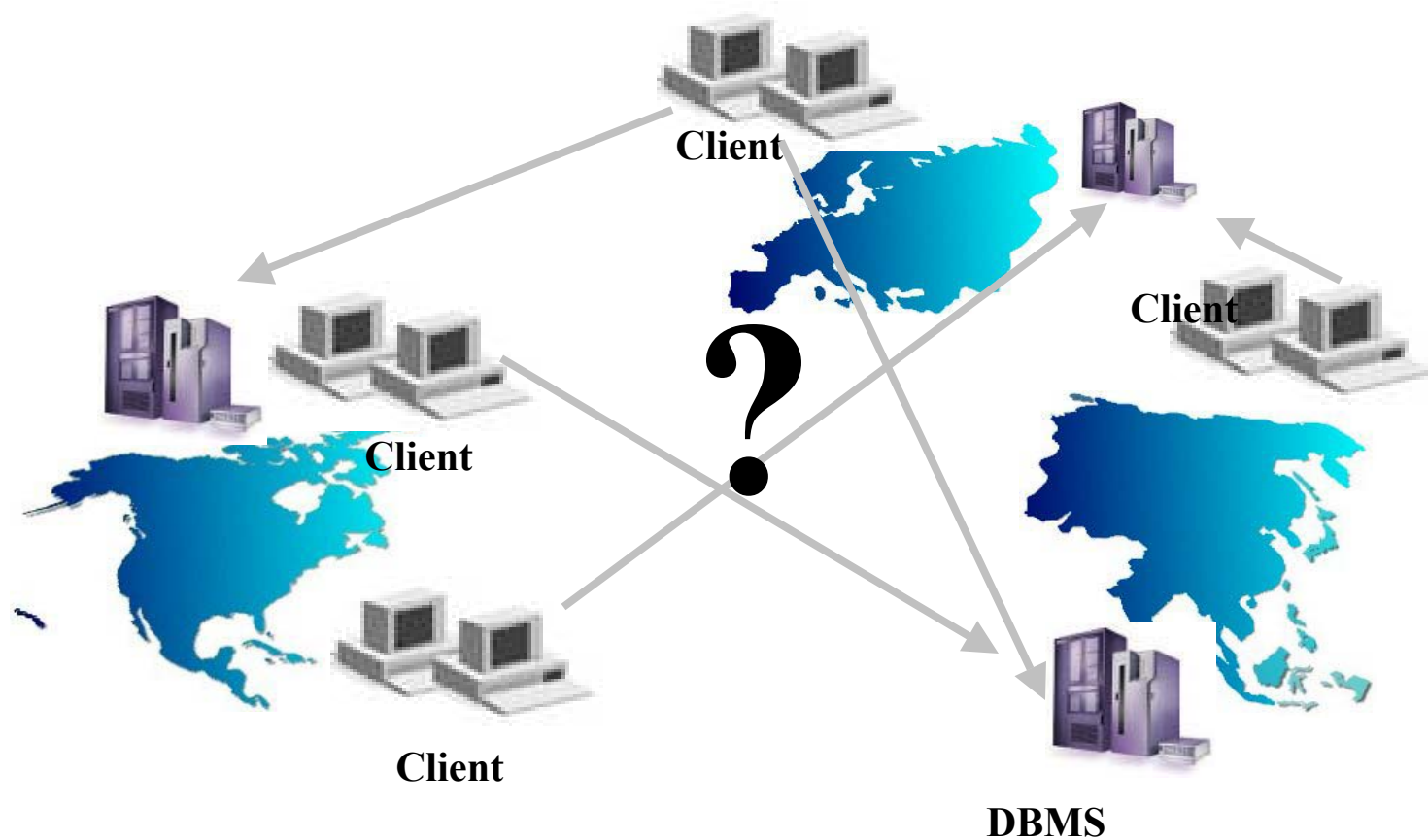
An easy way to manage Relational Databases  
in the Globus Community

**Sandro Fiore**

ISUFI/ Center for Advanced Computational Technologies  
Director: prof. Giovanni Aloisio  
University of Lecce, Italy

## A simple Scenario

*“How can Grid-aware Applications interact with their relational Data Resources in a distributed environment in order to make the most of a computational Grid?”*



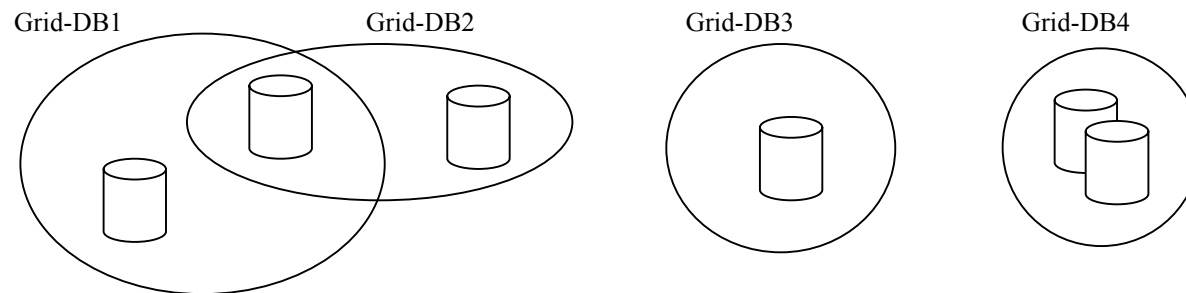


## Definition of Grid-DBMS

*A **Grid-DBMS** is a system which dynamically and transparently **reconfigures** components such as Data Resources at runtime, according to the Grid state, in order to maintain a desired performance level. It must offer an efficient, robust, intelligent, transparent, uniform access to **Grid-Databases***

## Definition of Grid-DataBase

*A **Grid-DataBase** is a collection of one or more Databases which can also be heterogeneous and contain replica, accessible through a Grid-DBMS front end . It represents an extension and a virtualization of the Database concept in a grid environment."*





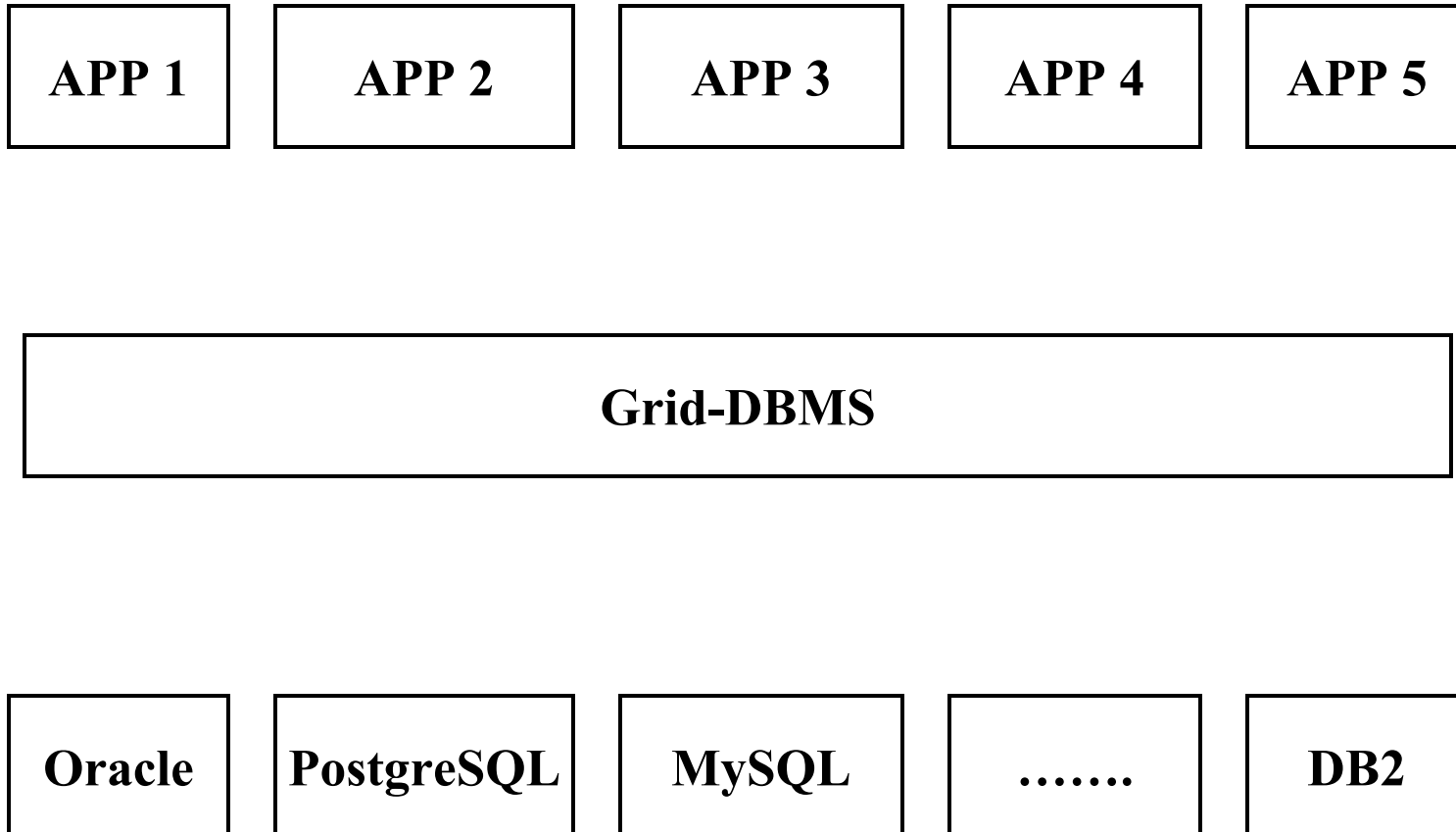
# Dynamic Reconfiguration

*What do we mean by Dynamic Reconfiguration?*

- *Dynamic Database Relocation*
- *Dynamic Database Replication*
- *Dynamic Database Partition*



# The **Grid-DBMS** layer





# Grid-DBMS requirements

## **A Grid-DBMS must be:**

- Secure
- Transparent
- Easy to manage
- Robust
- Efficient
- Intelligent

## **...and it must support:**

- Different DBMS
- High level functionalities
- High level Grid technologies(e.g. GridFTP)
- Dynamic reconfiguration mechanisms
- Performance Monitoring of the DBMS



# Introducing the GRelC Project

**Grid Relational Catalog** is a project that aims at designing and deploying the first **Grid-DBMS** for the **globus community**





# First Steps

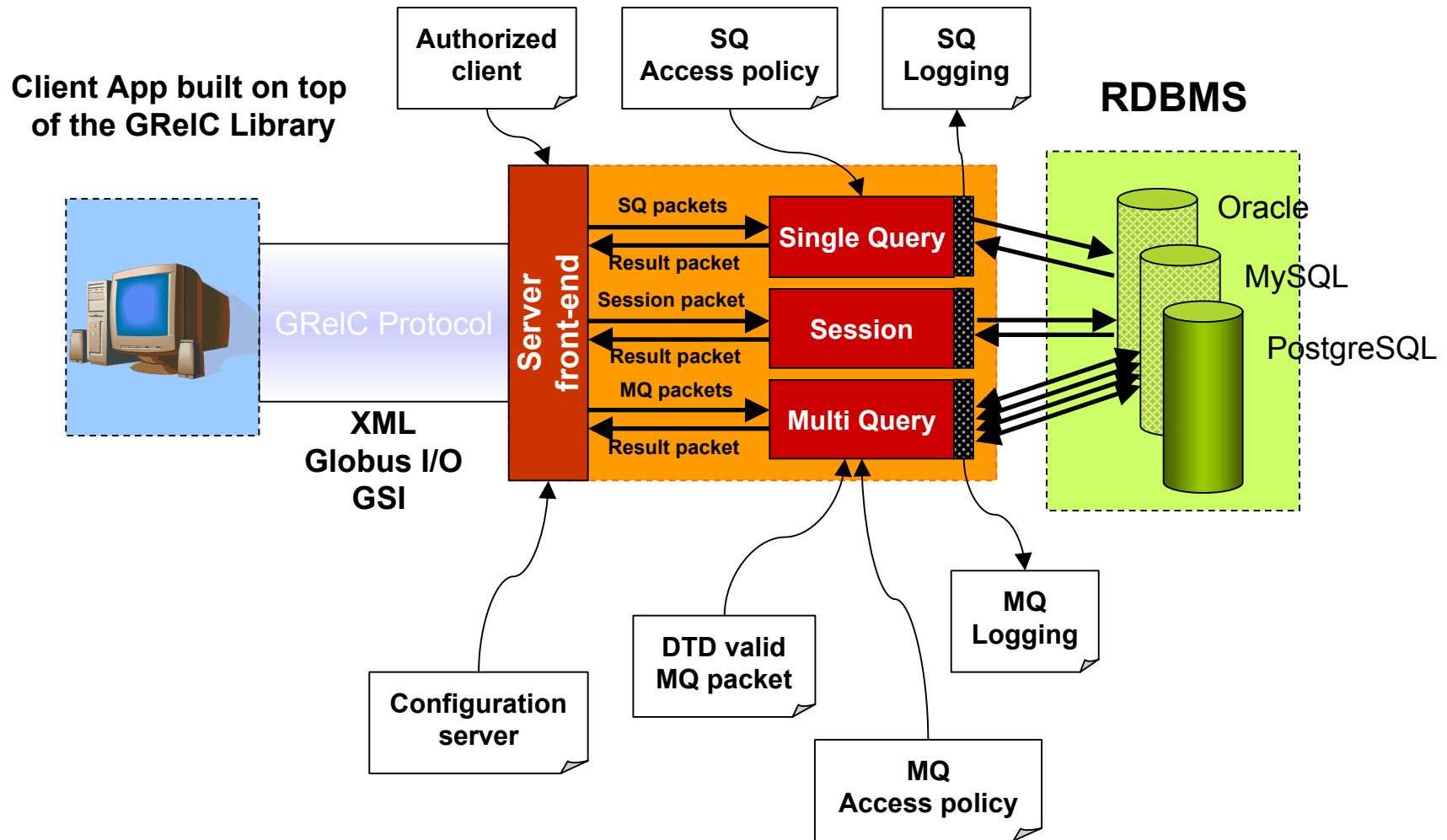
## **Connection**

- Drivers (basic building blocks)

## **Interaction**

- Queries (core and advanced)

# GReIC Basic Architecture





## Main Features

- Authentication
- Authorization
- Access control policy
- Data Encryption
- Single Query Support
- Multi Query Support
- MultiDBMS Support
- XML Data Validation
- Logging



# GRelC-Server Configuration

```

<CONFIGURATION_SERVER>
  <SERVER_PORT>13002</SERVER_PORT>
  <VALIDATION_DATA_STREAM>y</VALIDATION_DATA_STREAM>
  <REPOSITORY_DATA_PACKET>../grelc_repository_data_packet/</REPOSITORY_DATA_PACKET>
  <DATABASES>
    <DATABASE GRELC_DBNAME="Student">
      <DB_HOST_NAME>gandalf.unile.it</DB_HOST_NAME>
      <DB_NAME>grelcdb</DB_NAME>
      <DB_LOGIN>db-login</DB_LOGIN>
      <DB_PASSWORD>db-pwd</DB_PASSWORD>
      <DB_PORT>5432</DB_PORT>
      <DTD_FILENAME>../grelc_dtd/grelc_schema2.dtd</DTD_FILENAME>
      <AUTHORIZATION_CLIENT>y</AUTHORIZATION_CLIENT>
      <AUTHORIZED_CLIENT>
        <DN INSERT="TRUE" DELETE="FALSE">DN-user1</DN>
        <DN CREATE_DB="TRUE" DROP_DB="TRUE">DN-user2</DN>
        <DN UPDATE="TRUE" GRIDFTPSQ="TRUE">DN-user3</DN>
        <DN MQ="TRUE" INSERT="TRUE">DN-user3</DN>
      </AUTHORIZED_CLIENT>
    </DATABASE>
    ...
  </DATABASES>
</CONFIGURATION_SERVER>

```

**GRelC-Server General Info** {

**Database Student Configuration** {

**Database Authorization Policy** {

**Database Library Configuration** {

**Access Control Policy**



# Access Policy

<b>Access Policy</b>	<b>If true...</b>
CREATE_DB	Allow user to create new databases
DROP_DB	Allow user to drop databases
MQ	Allow user to do MultiQuery
GRIDFTPMQ	Allow user to do MultiQuery Grid FTP
GRIDFTPSQ	Allow user to do SingleQuery Grid FTP
TRANSACTION	Allow user to do transactions
INSERT	Allow user to do Insert Query
UPDATE	Allow user to do Update Query
DELETE	Allow user to do Delete Query



# Logging

## **GRelC\_Connection.log**

Connection from /O=Grid/O=Globus/OU=unile.it/CN=Sandro Fiore to **grelcdb** at 15/07/2003 13:25 [ OK ]

Connection from /O=Grid/O=Globus/OU=unile.it/CN=Daniele Lezzi to **grelcdb** at 15/07/2003 13:40 [ OK ]

Connection from /O=Grid/O=Globus/OU=unile.it/CN=Marco Polo to **grelcdb** at 15/07/2003 13:44 [ FAILED ]

## **GRelC\_server.log\_grelcdb**

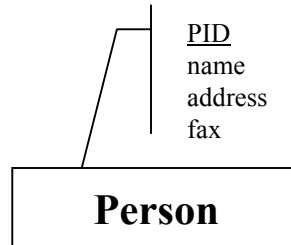
/O=Grid/O=Globus/OU=unile.it/CN=Sandro Fiore SINGLE select \* from student 15/07/2003 13:25

/O=Grid/O=Globus/OU=unile.it/CN=Sandro Fiore SINGLE select \* from seminar 15/07/2003 13:25

/O=Grid/O=Globus/OU=unile.it/CN=Daniele Lezzi SINGLE select title from seminar 15/07/2003 13:40



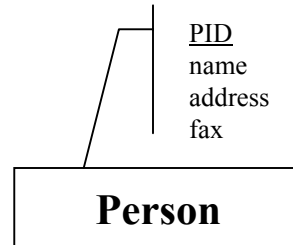
# MultiQuery XML file example



```
<TABLES GRELC_DBNAME="grelicdb">
  <TABLE NAME="person">
    <RECORDS>
      <RECORD>
        <ATTRIBUTES>
          <ATTRIBUTE NAME="PID" TYPE="STRING">DTJdfjksdk£23423</ATTRIBUTE>
          <ATTRIBUTE NAME="name" TYPE="STRING">Sandro Fiore</ATTRIBUTE>
          <ATTRIBUTE NAME="address" TYPE="STRING">Via Carlo V</ATTRIBUTE>
          <ATTRIBUTE NAME="fax" TYPE="STRING">+39 0832 297279</ATTRIBUTE>
        </ATTRIBUTES>
      </RECORD>
      <RECORD>
        <ATTRIBUTES>
          <ATTRIBUTE NAME="PID" TYPE="STRING">kjgjkgd£32424</ATTRIBUTE>
          <ATTRIBUTE NAME="name" TYPE="STRING">Marco Polo</ATTRIBUTE>
          <ATTRIBUTE NAME="address" TYPE="STRING">Via America</ATTRIBUTE>
          <ATTRIBUTE NAME="fax" TYPE="STRING">+39 0832 555777</ATTRIBUTE>
        </ATTRIBUTES>
      </RECORD>
    </RECORDS>
  </TABLE>
</TABLES>
```



# MultiQuery DTD file example

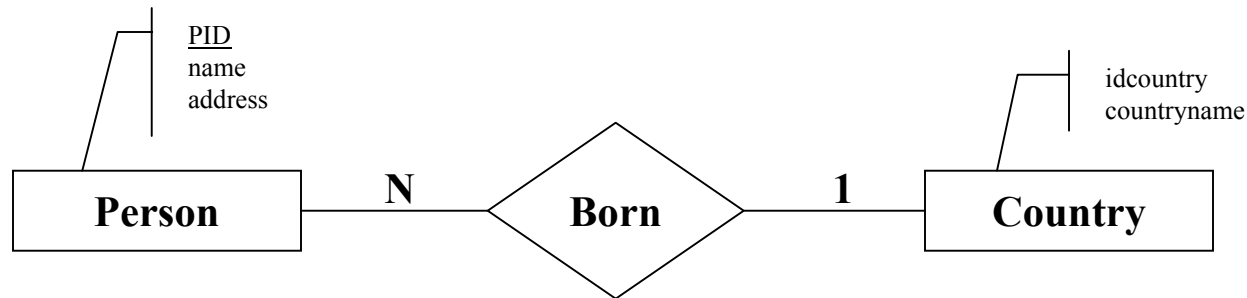


```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT TABLES (RELATIONS?,TABLE+)>
<!ELEMENT RELATIONS (RELATION+)>
<!ELEMENT RELATION (REFERENCEFIELDS)>
<!ELEMENT REFERENCEFIELDS (ATTRIBUTE+)>
<!ELEMENT TABLE (RELATIONS?,RECORDS)>
<!ELEMENT RECORDS (RECORD+)>
<!ELEMENT RECORD (ATTRIBUTES?, RELATIONS?)>
<!ELEMENT ATTRIBUTES (ATTRIBUTE+)>
<!ELEMENT ATTRIBUTE (#PCDATA)>
<!ATTLIST TABLES GRELC_DBNAME (grelcdb) #REQUIRED>
<!ATTLIST TABLE NAME (person) #IMPLIED>
<!ATTLIST ATTRIBUTE NAME ( name| PID |address | fax ) #IMPLIED
TYPE (INTEGER | FLOAT | DOUBLE | STRING | LONG) #IMPLIED>
```





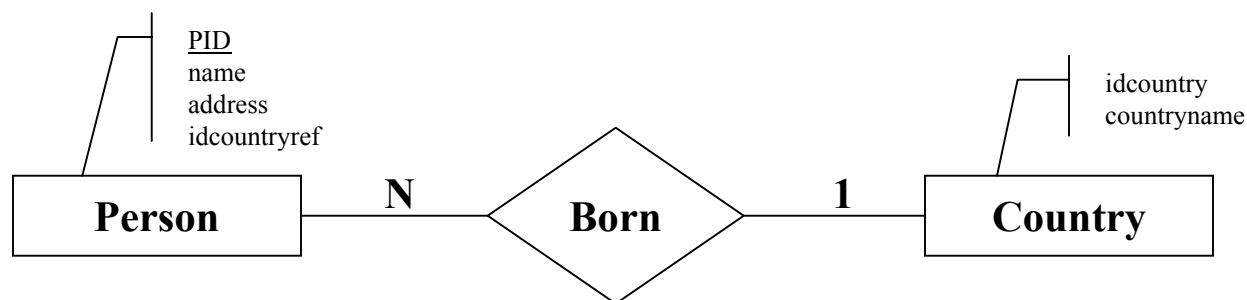
# MultiQuery XML file example



```

<TABLES GRELC_DBNAME="grelcdb">
  <TABLE NAME="person">
    <RECORDS>
      <RECORD>
        <RELATIONS>
          <RELATION FOREIGNKEY="idcountryref" REFERENCETABLE="country" REFERENCEKEY="idcountry">
            <REFERENCEFIELDS>
              <ATTRIBUTE NAME="countryname" TYPE="STRING">Italy</ATTRIBUTE>
            </REFERENCEFIELDS>
          </RELATION>
        </RELATIONS>
        <ATTRIBUTES>
          <ATTRIBUTE NAME="name" TYPE="STRING">Sandro Fiore</ATTRIBUTE>
          <ATTRIBUTE NAME="address" TYPE="STRING">Via Carlo V</ATTRIBUTE>
          <ATTRIBUTE NAME="PID" TYPE="STRING">jhdhsfdhj9833</ATTRIBUTE>
        </ATTRIBUTES>
      </RECORD>
    </RECORDS>
  </TABLE>
</TABLES>
  
```

## MultiQuery DTD file example



```

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT TABLES (RELATIONS?,TABLE+)>
<!ELEMENT RELATIONS (RELATION+)>
<!ELEMENT RELATION (REFERENCEFIELDS)>
<!ELEMENT REFERENCEFIELDS (ATTRIBUTE+)>
<!ELEMENT TABLE (RELATIONS?,RECORDS)>
<!ELEMENT RECORDS (RECORD+)>
<!ELEMENT RECORD (ATTRIBUTES?, RELATIONS?)>
<!ELEMENT ATTRIBUTES (ATTRIBUTE+)>
<!ELEMENT ATTRIBUTE (#PCDATA)>
<!ATTLIST TABLES GRELC_DBNAME (grelcdb) #REQUIRED>
<!ATTLIST TABLE NAME (person | country) #IMPLIED>
<!ATTLIST RELATION
FOREIGNKEY (idcountryref) #IMPLIED
REFERENCETABLE (country) #IMPLIED
REFERENCEKEY (idcountry) #IMPLIED>
<!ATTLIST ATTRIBUTE
NAME ( name | address | PID | countryname | idcountry ) #IMPLIED
TYPE (INTEGER | FLOAT | DOUBLE | STRING | LONG) #IMPLIED>
    
```

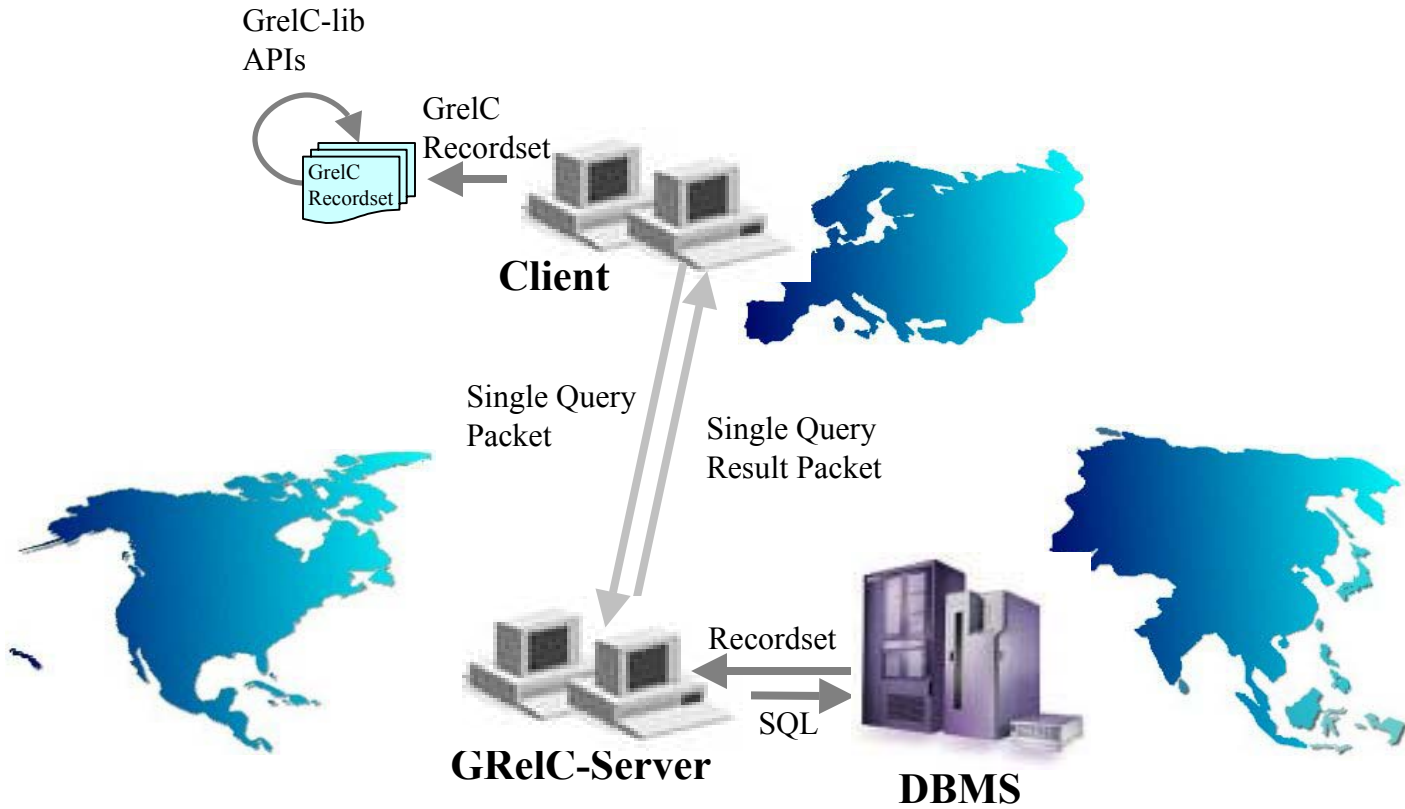


# GReIC QUERIES

You can submit several GReIC-Queries to the GReIC-Server:

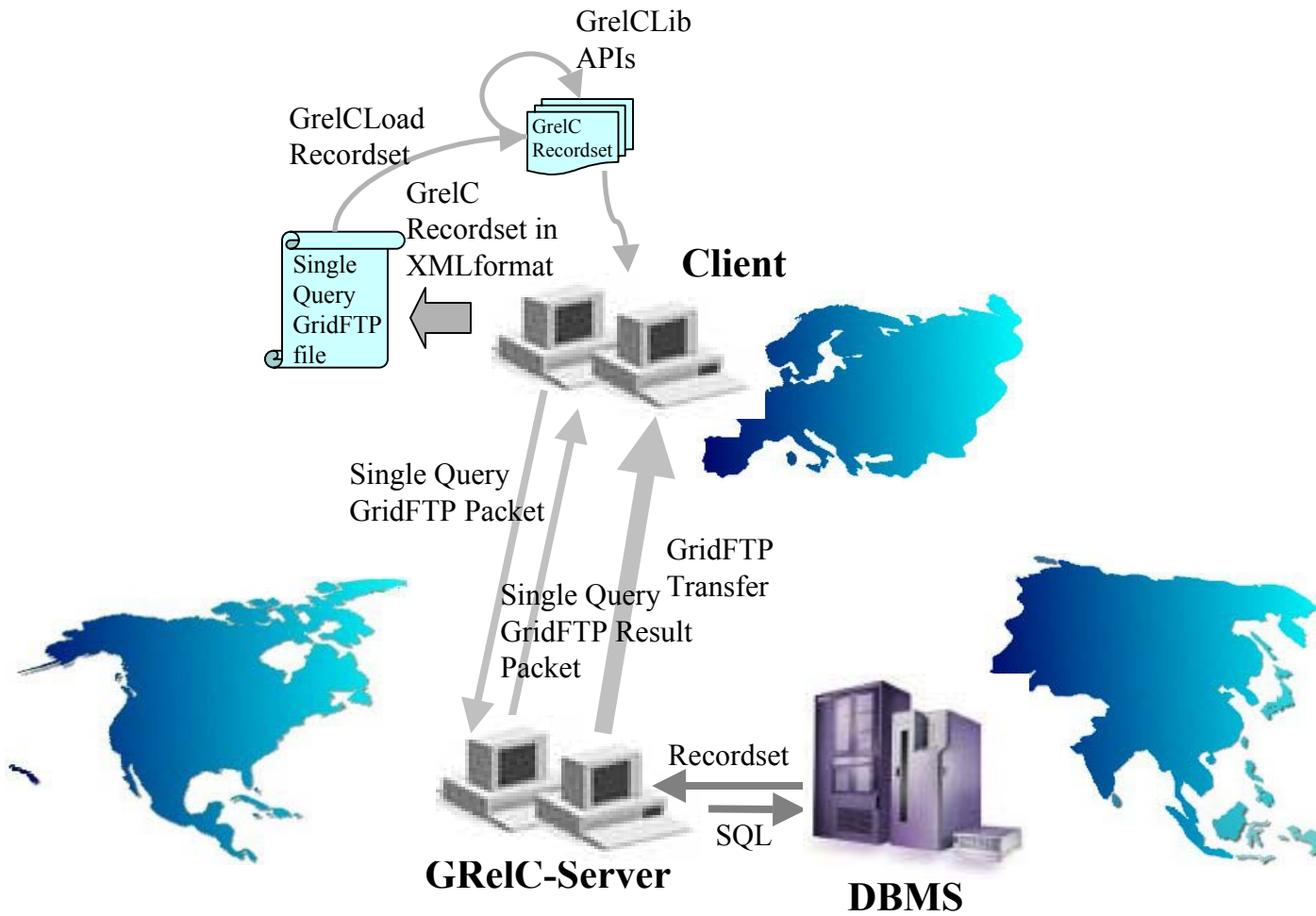
- 1) Single Query (*SQ*)
- 2) Single Query GridFTP (*SQ-GridFTP*)
- 3) Single Query Remote GridFTP (*SQR-GridFTP*)
- 4) Multi Query (*MQ*)
- 5) Multi Query GridFTP (*MQ-GridFTP*)
- 6) Multi Query GridFTP-ThirdParty (*MQ-GridFTP-TP*)

# SINGLE QUERY



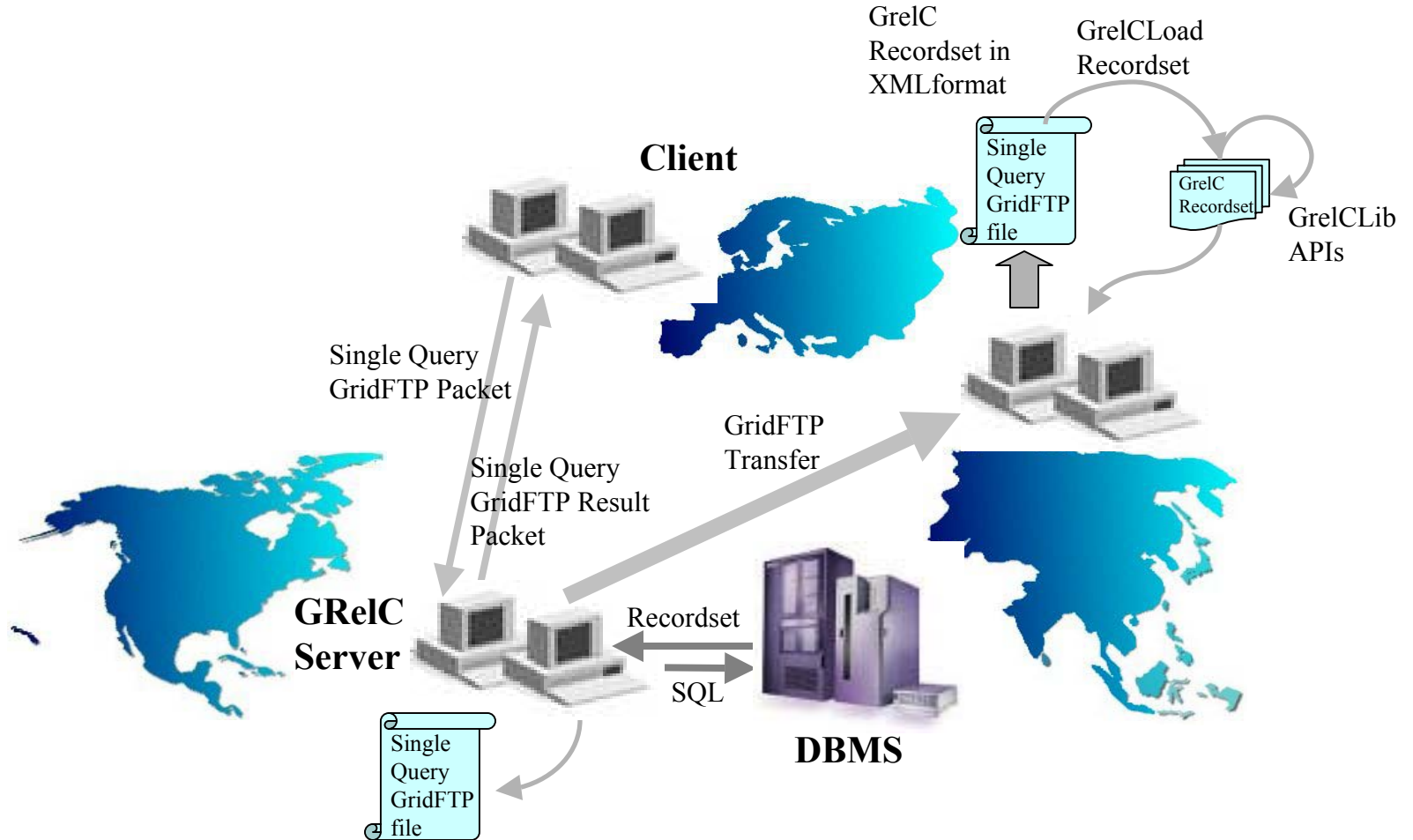


# SINGLE QUERY GRIDFTP

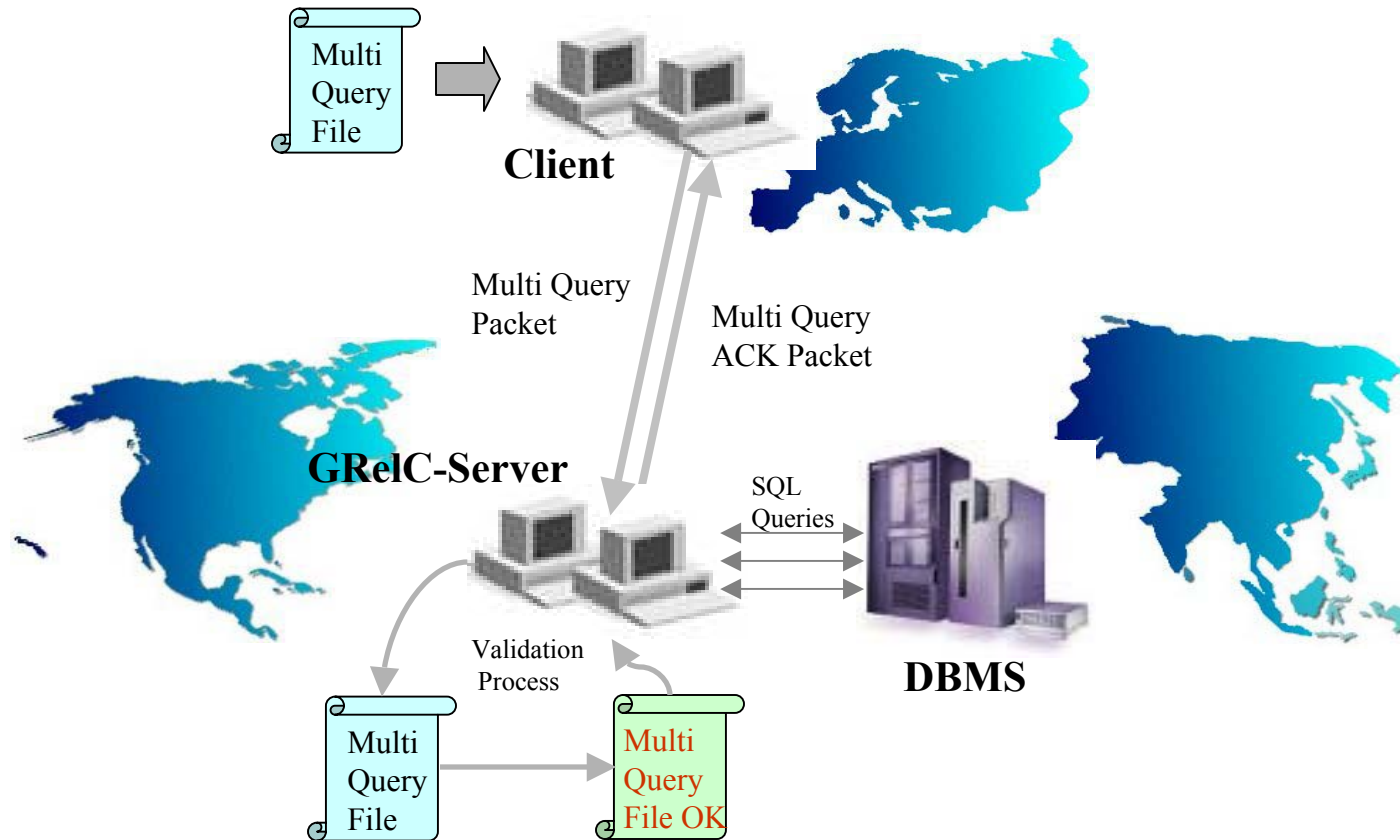




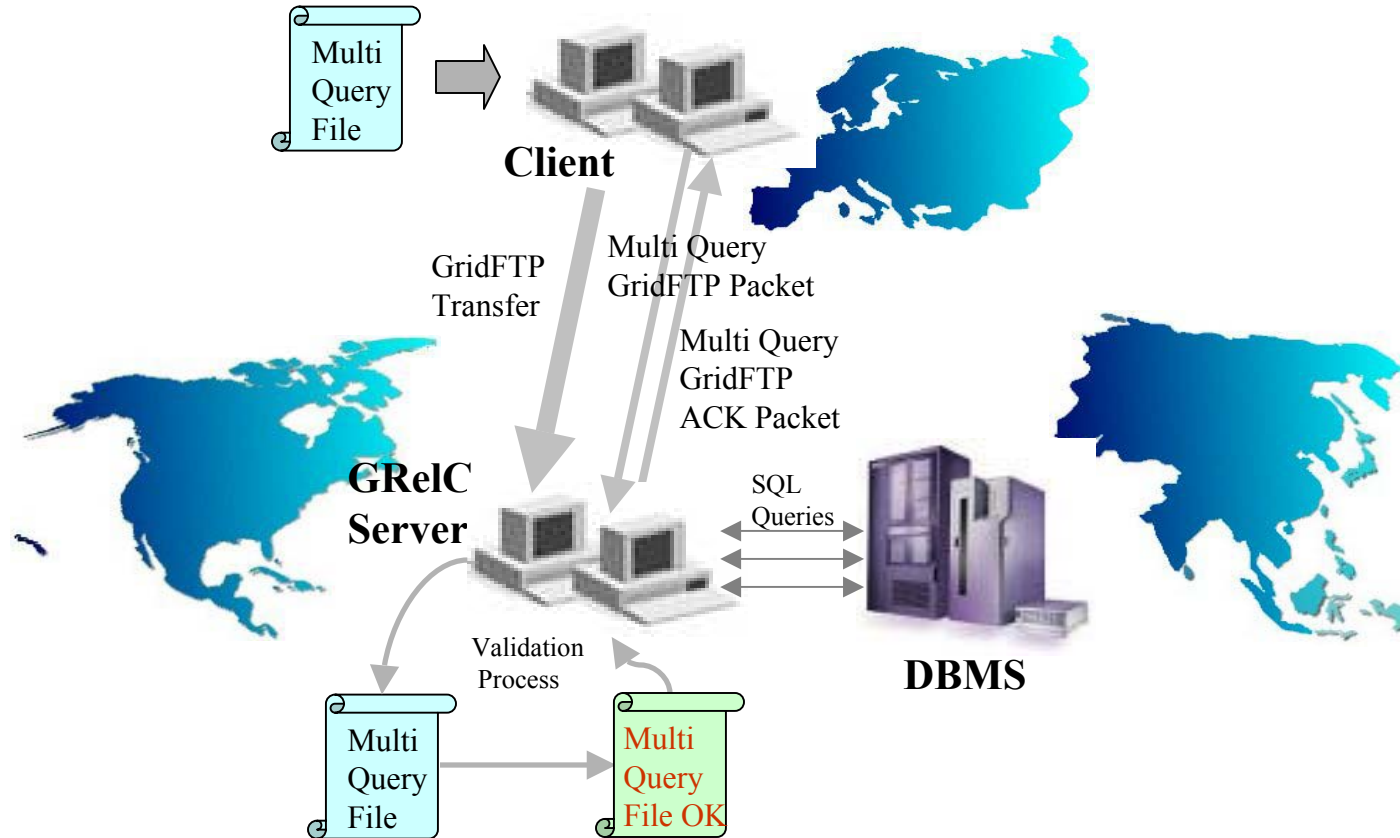
# SINGLE QUERY REMOTE GRIDFTP



# MULTI QUERY

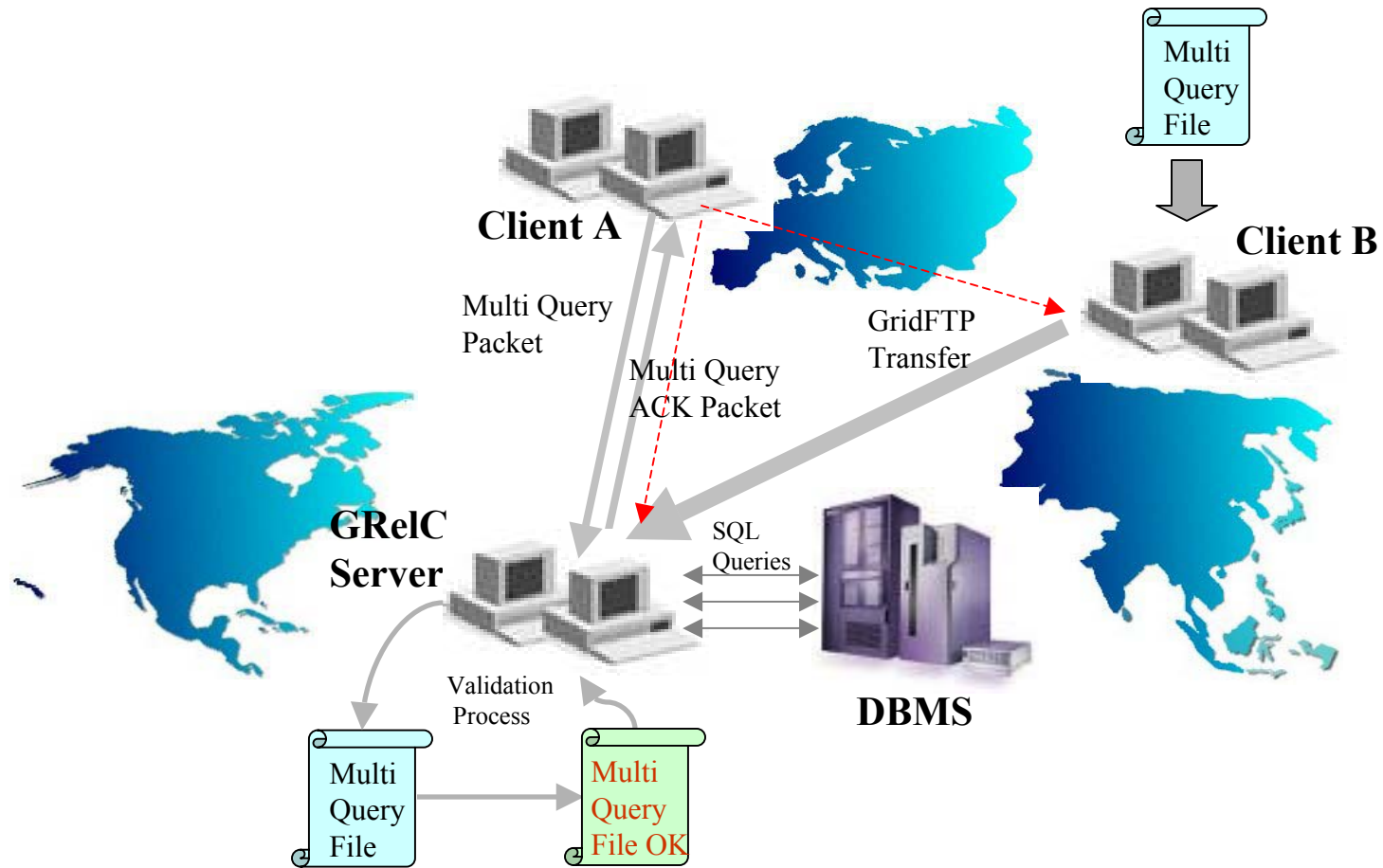


# MULTI QUERY GRIDFTP





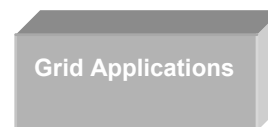
# MULTI QUERY GRIDFTP THIRD-PARTY





# The GRelC Library: a new layer

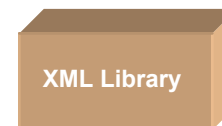
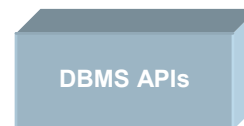
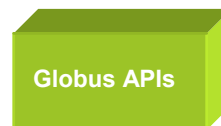
**Appl. built on top  
of the GRelC-lib**



**New Layer**



**Core Libraries**





# The GRelC Library: APIs Classification

We can classify the 42 APIs into 5 categories:

- 1) Connection APIs
- 2) Data Manipulation APIs
- 3) Core APIs
- 4) Administration APIs
- 5) High level APIs



## GRelC Library v2.0 (1/2)

```
int grelc_select(globus_io_handle_t*, char*, Grelc_Answer* );
int grelc_search_MQ(globus_io_handle_t*, char*, char* );
int grelc_grid_ftp_SQ(globus_io_handle_t*, char*, char*, char*, char* );
int grelc_grid_ftp_MQ(globus_io_handle_t*, char* );
int grelc_unbind(globus_io_attr_t* attr, globus_io_handle_t* handle);
int grelc_bind(globus_result_t result, char* hostname, unsigned short port, globus_io_attr_t* , globus_io_handle_t* );
int grelc_schema(Grelc_Answer* );
int grelc_schema_table(Grelc_Answer* );
int grelc_free_data(Grelc_Answer* );
void grelc_channel_initialization(globus_io_attr_t* attr,
    globus_io_secure_authorization_callback_t globus_io_secure_authorization_callback, void *args);
void grelc_channel_initialization_without_callback(globus_io_attr_t* attr);
int grelc_create_database(globus_io_handle_t* handle,char* database);
int grelc_drop_database(globus_io_handle_t* handle,char* database);
int grelc_create_table(globus_io_handle_t* handle,char* query);
int grelc_drop_table(globus_io_handle_t* handle,char* table);
int grelc_open_transaction(globus_io_handle_t* handle);
int grelc_abort_transaction(globus_io_handle_t* handle);
int grelc_rollback_transaction(globus_io_handle_t* handle);
int grelc_commit_transaction(globus_io_handle_t* handle);
```



## GRelC Library v2.0 (2/2)

```
int grelc_insert(globus_io_handle_t* handle,char* query);
int grelc_update(globus_io_handle_t* handle,char* query);
int grelc_delete(globus_io_handle_t* handle,char* query);
int grelc_get_number_records(Grelc_Answer* );
int grelc_get_number_fields(Grelc_Answer* );
int grelc_get_position_record(Grelc_Answer* );
int grelc_find_first(Grelc_Answer* data, char* attribute, char* comp, char* value);
int grelc_find_next(Grelc_Answer* data, char* attribute, char* comp, char* value);
int nomatch(Grelc_Answer* data);
int grelc_move_first(Grelc_Answer* );
int grelc_move_last(Grelc_Answer* );
int grelc_move_next(Grelc_Answer* );
int grelc_move_previous(Grelc_Answer* );
int grelc_move(Grelc_Answer* ,int );
int grelc_eof(Grelc_Answer* );
int grelc_bof(Grelc_Answer* );
int grelc_is_null(char* );
char* grelc_get_field_by_attribute(Grelc_Answer* ,char* );
char* grelc_get_field_by_position(Grelc_Answer* ,int );
char* grelc_get_name_field_by_position(Grelc_Answer* ,int );
```



# How to use the GRelC Library

```
Connection → { grelc_channel_initialization_without_callback(&attr);
                  grelc_bind(result,hostname,database_name,port,&attr,&handle);
Query → { grelc_select(&handle,query,&data);
Close Connection → { grelc_unbind(&attr,&handle);
Data Manipulation → { // Library Usage //
                        printf("Number of Records %d\n",grelc_get_number_records(&data));
                        printf("Number of fields %d\n",grelc_get_number_fields(&data));
                        grelc_move_first(&data);
                        while(!grelc_eof(&data)){
                            for (i=1; i<=grelc_get_number_fields(&data) ; i++)
                                printf("Field: %s ->
%s\n",grelc_get_name_field_by_position(&data,i),grelc_get_field_by_position(&data,i));
                                grelc_move_next(&data);
                            }
                        grelc_free_data(&data);
                        exit(EXIT_SUCCESS);
                    }
```



# Releases

Two current releases:

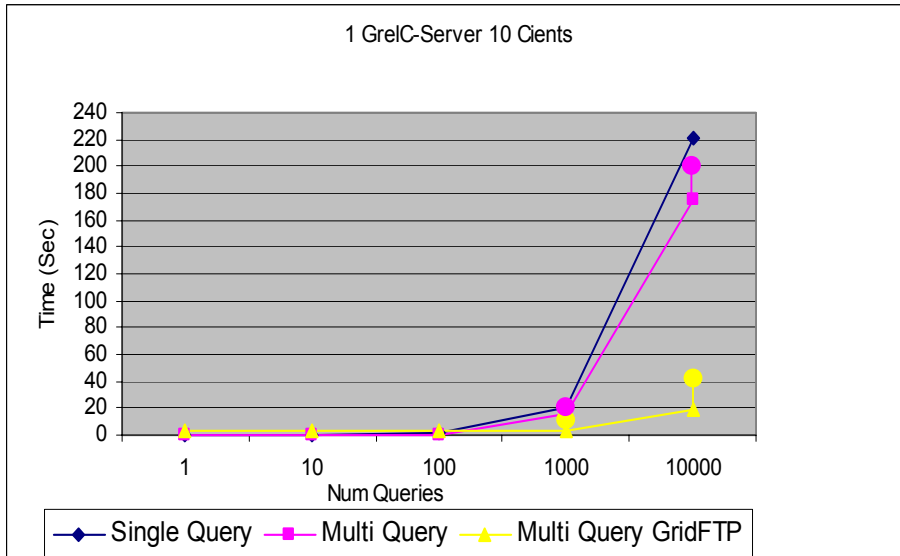
1. *GRelCv1.0*
2. *GRelCv2.0*

Differences:

- Only 23 APIs in the first version vs 42 in the second one.
- Different Grelc-Server management
- New operations for data manipulation
- Extended recordset structure
- Access control policy
- Logging
- High-level functionalities supported.



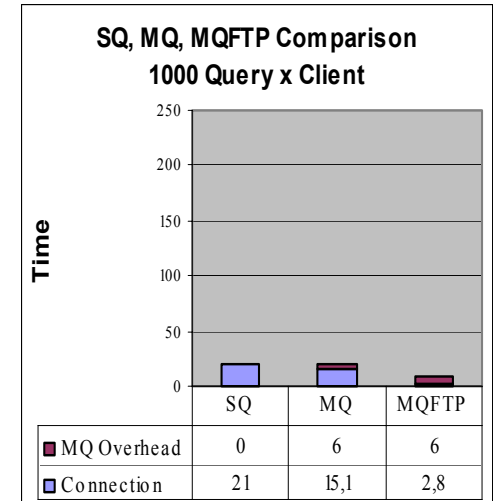
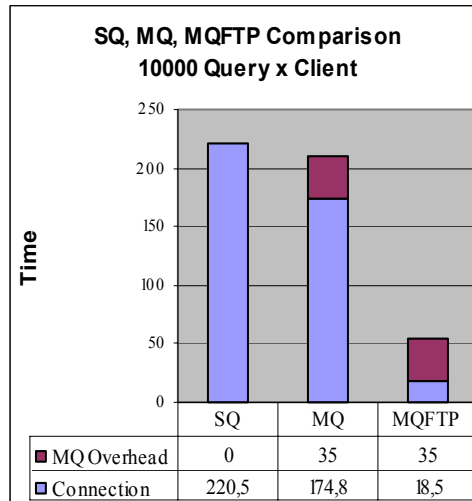
# First Tests in our Campus



10 Clients each one submitting n (1:10000) insert queries to a unique GRelC-Server  
(10:100.000 total insert queries)

Three different ways to do that.

- 1) 1:10.000 Single Query
- 2) 1 MultiQuery
- 3) 1 MultiQuery GridFTP







## Two parallel directions

- **GReIC Library** (more performant)

Industries

Real Applications

- **Web/Grid Services** (less performant but OGSA compliant)

Academic environment

Research



## Future Works

- **Web/Grid Services** Version (a basic version is already deployed and used for internal projects)
- Support for **Oracle, MySQL** DBMS
- Support for **Distributed Query** (very hard and interesting challenge)
- **Library Extensions** (new APIs)
- New Queries that support **compression** mechanisms
- **Scheduling** strategies related to replicated and partitioned databases
- **XML temporary datasets** management



## **For any information**

**Director:** Prof. Giovanni Aloisio (giovanni.aloisio @unile.it)

**Project P. I. :** Sandro Fiore (sandro.fiore@unile.it)

Center for Advanced Computational Technologies - CACT/ISUFI,  
University of Lecce - ITALY

**WebSite :** <http://gandalf.unile.it/grelc.html>