R-GMA
Now With Added Authorization

Steve Fisher on behalf of the R-GMA team

3rd EGEE Users Forum

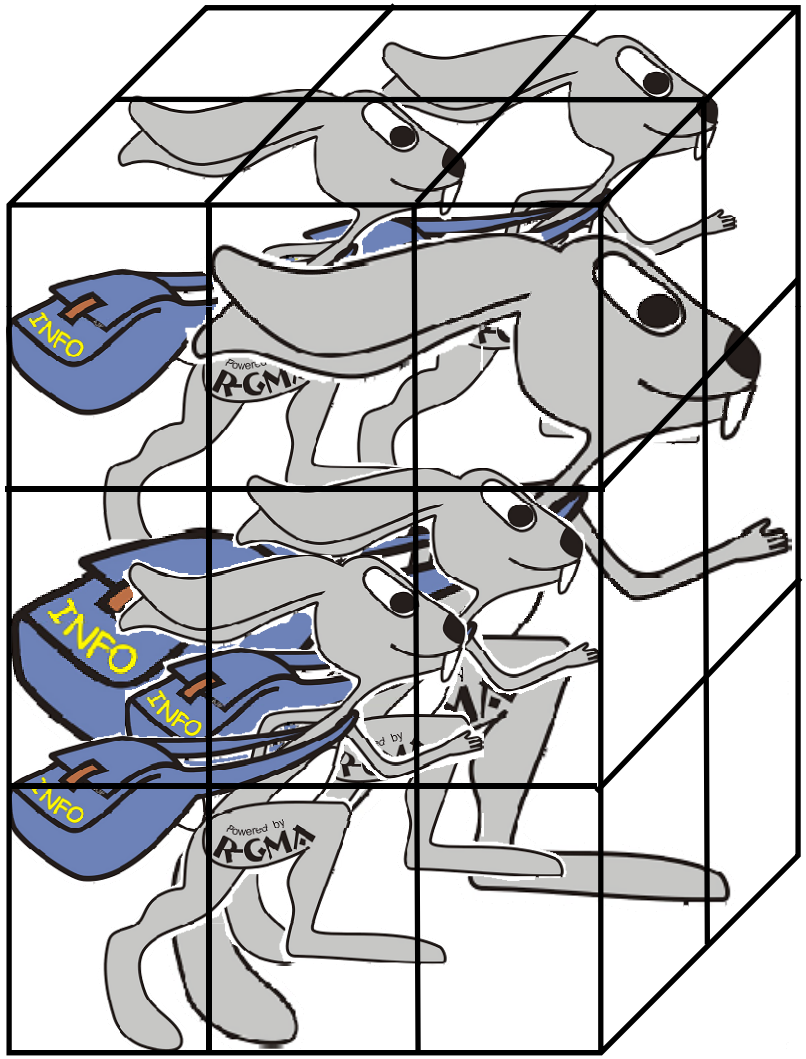
Le Polydôme, Clermont-Ferrand, France

www.eu-egee.org

Information Society and Media

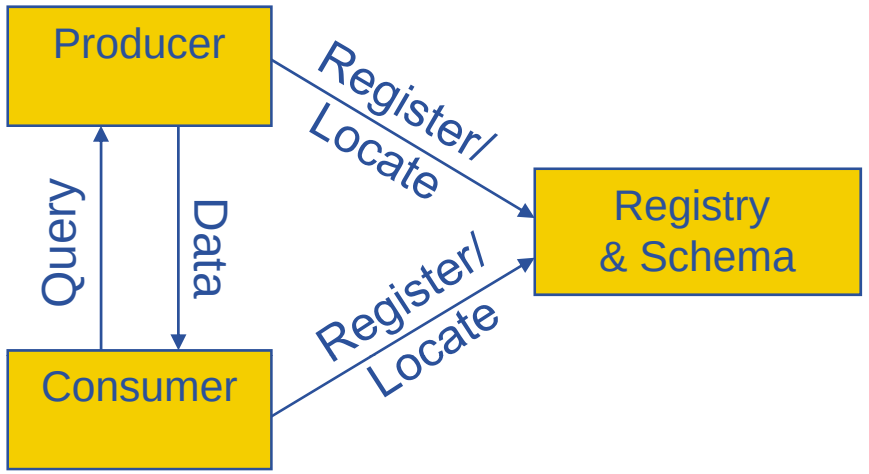
Overview

- **What it is**
- **New Design**
  - Engineered for robustness and scalability
- **New Features**
  - Orthogonality of producer type and of storage mechanism
  - Fine grained authorization
  - Multiple virtual databases
- **Status**
- **Summary**



R-GMA – what it is

- R-GMA is a distributed system for information and monitoring (following OGF's GMA)
- Users define their own data structures along with the fine grained authorization rules specifying who can write and read the data
- Users publish data via a producer API without knowledge of potential consumers
- A consumer API is used to retrieve the permitted view of information published by the producers

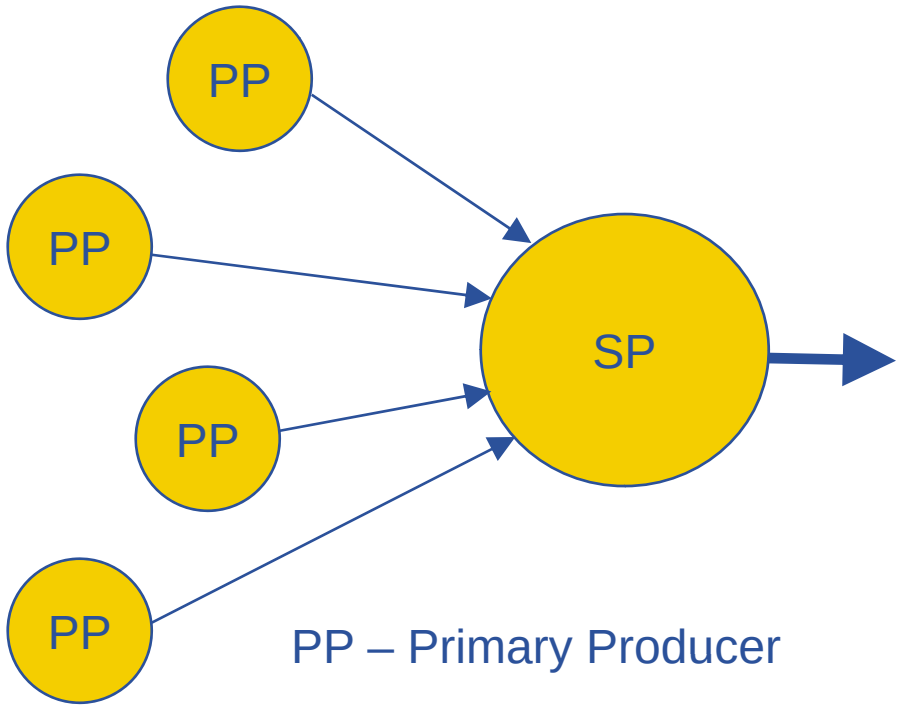


New Design

- **New design was motivated by:**
  - Problems seen in production
  - Need to add new features

New Design: Real SPs

- **Primary – source of data**
- **Secondary – republish data**
  - Co-locate information to speed up queries
  - Reduce network traffic

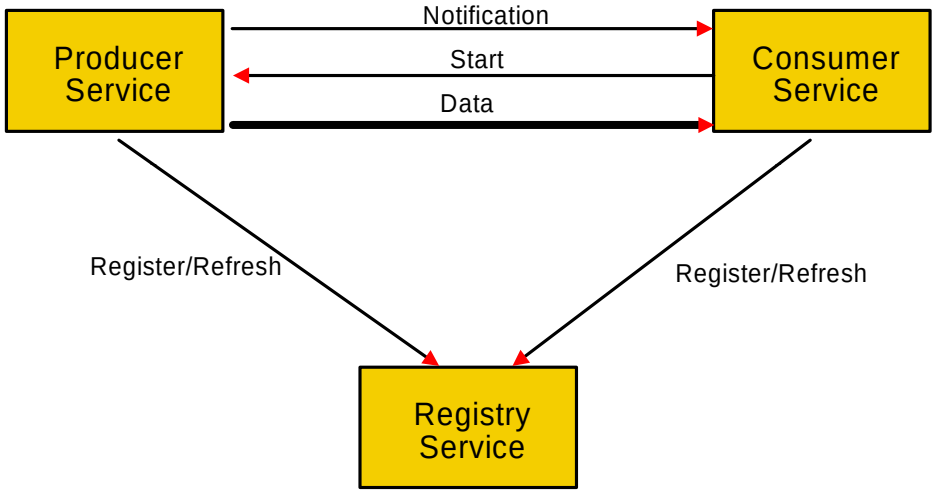


- SP is no longer constructed from a PP and multiple consumers but is a self-contained service – so now much more efficient
- Local calls on a MON box do not go through https returning XML then parsing it

New Design: Managing Memory Usage

- **May share servlet container (Tomcat) with other servlets**
- **JVM may be badly configured**
- **Use JDK 5's MXBeans to detect the low memory condition**
  - Solution should be portable across JVMs
- **When memory is low an RGMABusyException is returned for user calls that may take extra memory**
  - Inserting data into the system
  - Creating new producer or consumer resources

New Design: Control Messages



- For a producer a register message returns the consumers of interest and vice versa.
- Registration messages are re-sent periodically
- Reliance on the delivery of individual control messages has been removed
- Messages to other servers are wrapped in a task handled by our new Task Manager
- New messages supersede old ones
  - No build-up of queues
- Autonomy of services

New Design: Schema and Registry Replication

- **Schema**
  - Each server has full schema information
  - There is a master schema and all updates are first done on the master
  - Failure of the master would prevent schema updates but would have no impact upon producers or consumers

- **Registry**
  - We anticipate 2 or 3 registry instances
  - Server chooses registry instance to use based on response time
  - Server makes a new choice if existing instance does not respond

New Features

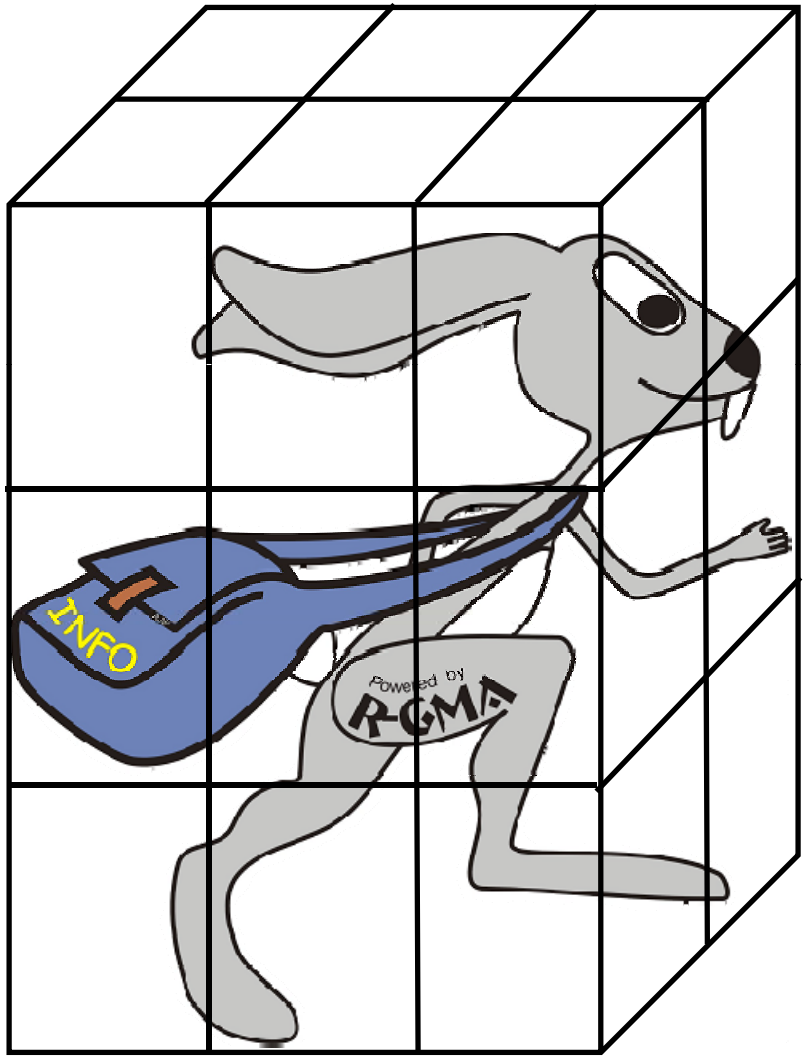
- Orthogonality of producer type and of storage mechanism
- Fine grained authorization
- Multiple virtual databases

New Features: Orthogonality of producer type and of storage mechanism

- **Previously:**
  - Memory
    - Continuous
  - Database
    - Latest or History
- **Now:**
  - Memory
    - Any combination of Continuous Latest and History
  - Database
    - Any combination of Continuous Latest and History

New Features: Fine Grained Authorization

- **Users can define their own fine grained authorization rules specifying who can write and read the data in each table element**
- **Rules stored in the schema – and can only be modified by the person creating and therefore owning the table**
- **Authorization is done using SQL views of tables constructed dynamically from:**
  - User defined rules
  - VOMS attributes



New Features: Fine Grained Authorization

- **Rules are added to grant access only (not to deny it) and they are cumulative - default is no access**
- **Rules have the form "predicate : credentials : action"**
  - The predicate defines the subset of rows of the table or view to which this rule grants access
  - The credentials define the set of credentials required for a user to be granted access to the subset of rows defined by this rule
  - The action defines what any matching user is allowed to do to the subset of rows defined by this rule (R, W or RW)
  - For example:
    - **::RW** grants full access to any authenticated user, to all rows in the specified table

Authorization Rules - Predicate

- **Predicate is an SQL WHERE clause comparing the values in specified columns with constants, other columns or credential parameters (credential name in square brackets, such as [DN]) that are replaced by the corresponding credentials from the user's certificate**
- **This clause may be empty, in which case the rule applies to all rows in the table**
- **For example:**
  - WHERE Owner = [DN] Selects rows where the "Owner" column matches the DN on the certificate

Authorization Rules - Credentials

- **Credentials is a boolean combination of equality constraints of the form [credential] = constant**
- **May be empty, in which case the rule applies to all authenticated users**
- **For example:**
  - [GROUP] = ’Marketing’ OR [GROUP] = ’Management’ states to which VOMS groups the rule applies

Authorization Examples

```
WHERE Section = 'Marketing':[GROUP] = 'Marketing' OR [GROUP]
   = 'Management':RW
```

- Grants read-write access to any authenticated user with a GROUP credential of 'Marketing' or 'Management', to those rows that contain the value 'Marketing' in the 'Section' column

```
WHERE Owner = [DN]::R
```

- Grants read-only access to any authenticated user, to those rows that contain the value of their DN credential in the 'Owner' column

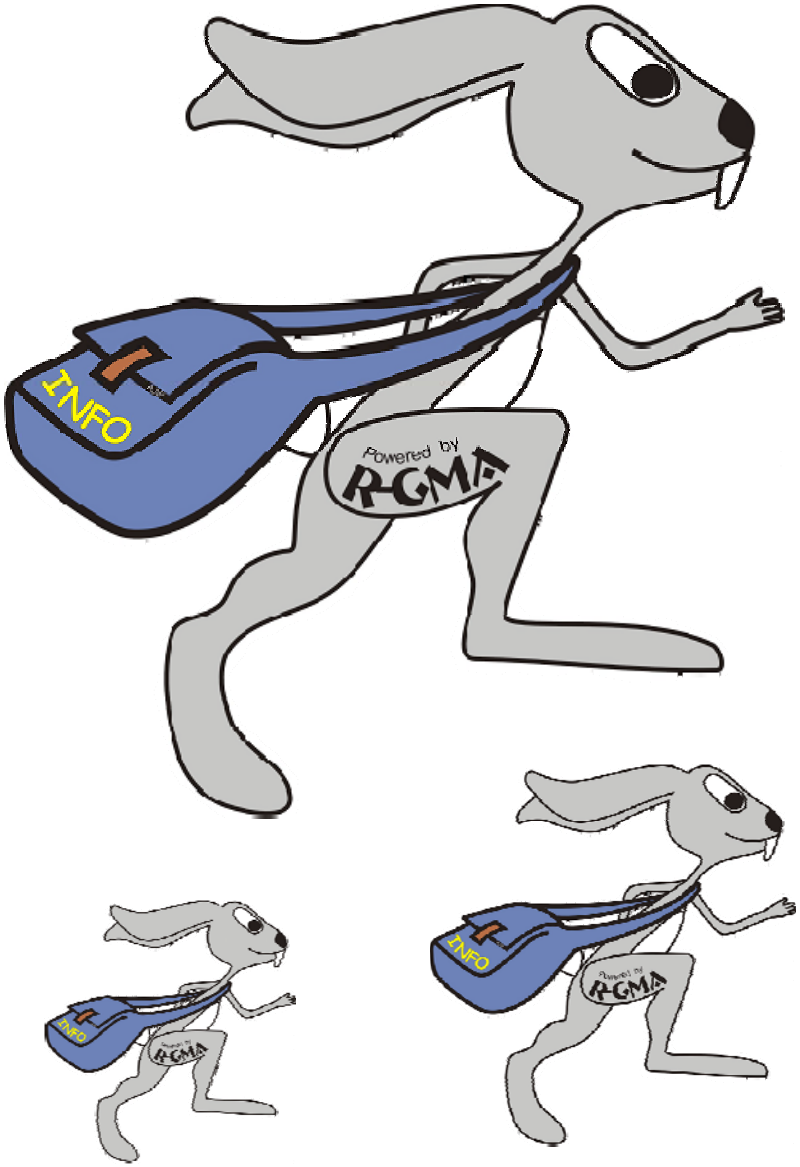
```
WHERE Group = [GROUP] OR Public = 'true'::R
```

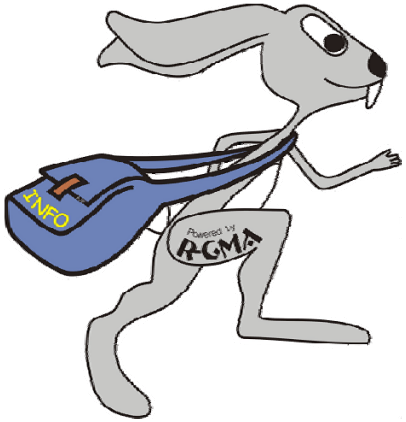
- Grants read-only access to any authenticated user, to those rows that contains one of their GROUP credential values in the 'Group' column, or have a value of 'true' in the 'Public' column

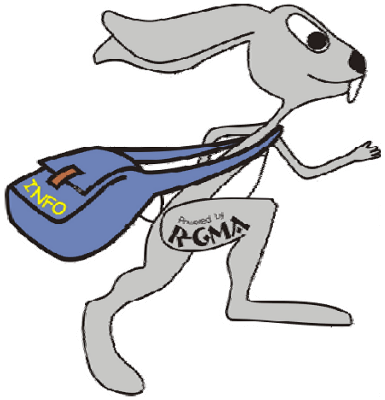
```
::R
```

- Grants read-only access to any authenticated user, to all rows in the table

New Features: Virtual Databases



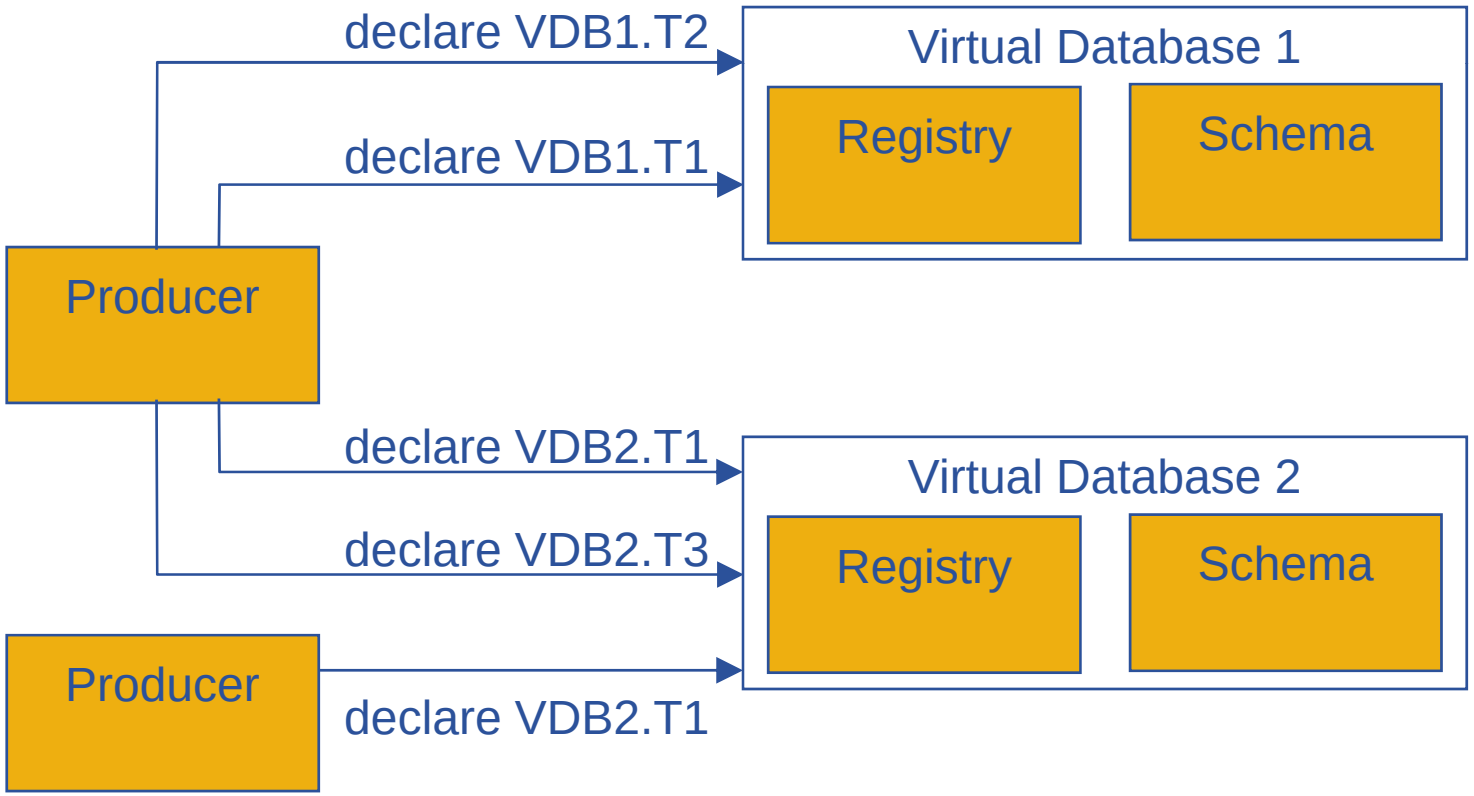




- **One R-GMA schema for the whole world is not scalable**
- **Introduce VDB as a namespace mechanism**
- **We expect that a VO will define one or more VDBs**
  - Will phase out the “default” VDB
- **Each VDB will have:**
  - Several registry replicas
  - A schema replica at each site supporting that VDB
  - One schema defined as the master schema for each VDB

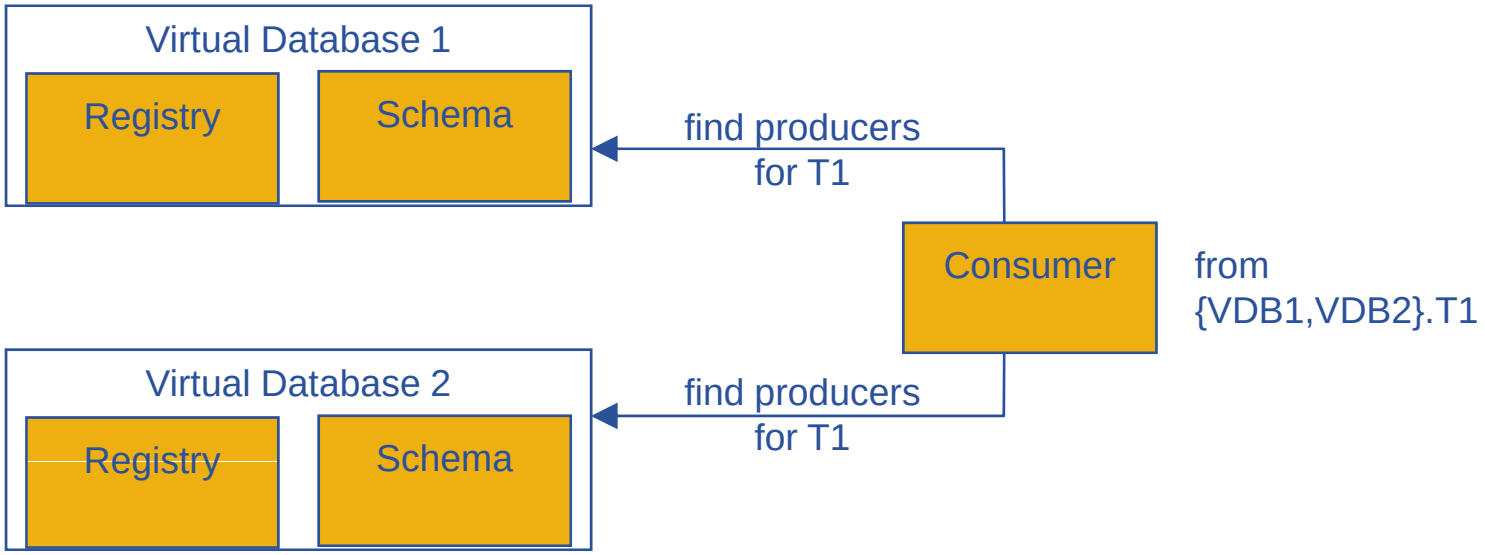
Publishing to Multiple VDBs

- **When data are inserted into a producer, the data are published into a specific VDB**
- **It is possible for a producer service to publish to more than one VDB**



Querying Multiple VDBs

- **Queries may be evaluated over several VDBs**
- **Normal SQL syntax of a database prefix before the table name is used to specify the VDB**
- **SQL joins across tables in multiple VDBs are supported**
- **A special union syntax has been defined: "SELECT * FROM {VDB1,VDB2}.T" to indicate that the query should be evaluated over the union of the tuples from table T in VDB1 and VDB2**
  - It requires that the tables T from the two VDBs are identical



Status

- All components written and a lot of testing has already been done
- More testing being done
- Expected to be offered to certification around end of the month

This will give us a highly reliable, functional and scalable R-GMA

Summary

- **From the existing deployment we learned:**
  - Firewalls do get reconfigured
  - Users do the most unexpected things
- **New design:**
  - We have tried to think of everything that can go wrong
  - Made the system self correcting and avoided critical messages
  - Have introduced the task manager
  - Have provided schema and registry replication
- **New Features:**
  - Multiple virtual databases (VDBs) to partition the data
  - Users can define their own fine grained authorization

With thanks to: Information Society and Media, GridPP UK Computing for Particle Physics, Trinity College Dublin, Coláiste na Tríonóide, Baile Átha Cliath, The University of Dublin