

Developer and User Manual

Database Browser Portlet V.0.3

Table of Contents

1	Introduction	4
1.1	Data Access Services (DAS)	4
1.2	Database Browser Portlet	5
1.2.1	Objective	5
1.2.2	Implementation	6
1.2.3	How it works	7
2	Getting Started	9
2.1	Requirements	9
2.2	Getting the Files	9
2.3	Installing Database Browser Portlet	9
2.3.1	Deploy the WAR	9
2.3.2	Deploy the Source	10
2.4	Operating the Database Browser	11
2.4.1	Browsing accessible database	12
2.4.2	Getting the table schema	12
2.4.3	Getting the table contents	13
2.4.4	Handling the data	14
2.5	Executing Distributed SQL queries	16
3	References	19

List of Figures

Figure 1: Higher-level Architecture of DAS	4
Figure 2 Database Browser Portlet.....	6
Figure 3: Interactions between Portal, DAS and Shibboleth.....	7
Figure 4: Sequence diagram for the interactions	8
Figure 5 add the portlet in the desired layout	10
Figure 6 Database Browser GUI	11
Figure 7 Data Resources Panel.....	11
Figure 8 Table Panel.....	12
Figure 9 Table Schema Panel.....	12
Figure 10 Table Contents Panel	13
Figure 11 Sorting the Table	14
Figure 12 XML Window	14
Figure 13 CSV Window	15
Figure 14 DAS Query Tool for Single Query.....	16
Figure 15 DAS Query Tool for Multiple Query	17
Figure 16 Building the Multiple-Queries-Tree.....	18

1 Introduction

1.1 Data Access Services (DAS)

In ViroLab the Data Access Services (DAS) are designed and implemented to realize unified access to the distributed and heterogeneous data resources, which belong to different organizations within the laboratory environment. It acts as Single/central entry point for all data requests and represents the only “visible” and accessible system that hide all communication and transformation activities from the users and developers respectively[D2.3].

The DAS provide standard web service interfaces allowing the smooth invocation of single services’ capabilities. As shown in figure 1, DAS is composed of a set of virtualization services. The four main stand-alone service modules of DAS are Security handling, notification handling, storage handling and data handling. They provide functionalities for dealing with several heterogeneous databases concurrently, each serving a different purpose within the overall services’ infrastructure, have been developed independently to guarantee a certain level of flexibility, scalability, and sustainability by following the general approaches of the so-called Service Oriented Architecture (SOA) standards. For more detailed information about DAS please read the paper [DA07]

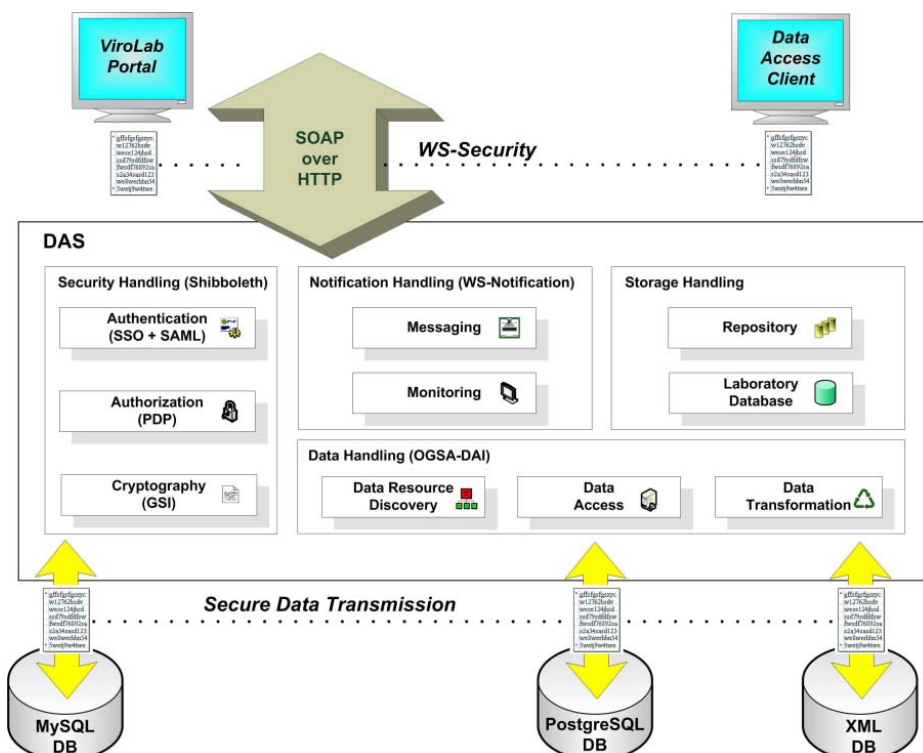


Figure 1: Higher-level Architecture of DAS

1.2 Database Browser Portlet

1.2.1 Objective

DAS enable two types of access into the virtual laboratory infrastructure.

1. Via logging into the working environment through a web portal.
2. Via the Data Access Client (DAC) being part of the Experimental Planning Environment (EPE).

Web portal implementation for ViroLab is based on a popular open-source Portlet-based web portal framework – GridSphere[GS]. The Portal is responsible for redirection to the proper Home Organization, display of the application portlets, and then requesting the chosen resources. In order to realize single sign on (SSO), the ViroLab Portal integrates GridSphere with Shibboleth[SBL]. When user selects his Home Organization from WAYF portlet of ViroLab Portal, he or she is redirected to login page of his or her HO. When he or she is authenticated properly, he or she is redirected together with the attributes encoded in SAML header to the portal login page. This login portlet is responsible for copying attributes including the user handle from portlet session to Portal level user session which makes them available to all portlets in the portal.

To achieve the first access possibility to DAS through the portal Database Browser Portlet for GridSphere is implemented. It provides ViroLab end-users, database administrators and mainly technicians (application developers) with an easily accessible and convenient facility for database inquiry and result management through the centralize ViroLab portal with help of its graphical and user-friendly interface (current version is v0.2):

- Authorize ViroLab users to access the appropriate database by using the Authentication module in DAS, which integrates with Shibboleth's Service Provider features. The portlet gets user identity (Handle) and IDP address from session inside portal after the already done authentication for authorization.
- Browse accessible databases in search for desired data of patents, drug, mutations etc. and their released schema definitions
- Perform single or multiple distributed SQL queries to single or multiple Databases. An individual database is provided to facilitate the store, read and perform SQL queries.
- Grant functions for searching for precise data in results of performed queries, sorting data in different order, and printing, saving data as xml, html or csv.

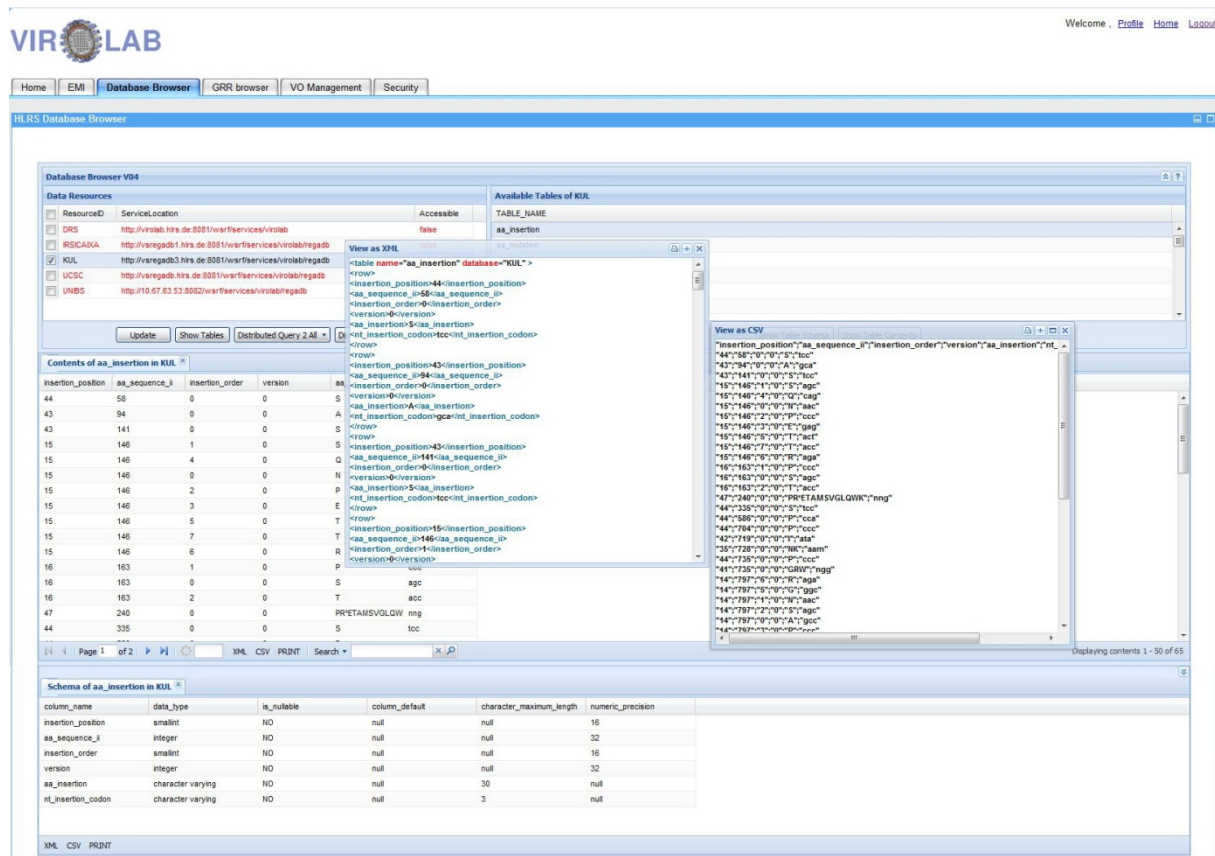


Figure 2 Database Browser Portlet

1.2.2 Implementation

Database Browser portlet is essentially an implementation of the JSR 168 portlet API [JSR] standard for the GridSphere portal framework, which one is integrated with Shibboleth to enable single sign on. The portlet and the portal reside in presentation layer of the Virtual Laboratory. Database Browser portlet is empowered by AJAX to build the client-side graphical and user-friendly interface and server-side rpc services for accessing DAS. An AJAX portlet enable us to handle great amount of data with decrease in bandwidth use, almost no refresh delay and a practical and convenient AJAX GUI.

Thanks to the powerful Google Web Toolkit [GWT] of Google and GWT-Ext [GWTE], we did write our AJAX front-end easily and effectively in the Java programming language which GWT then cross-compiles into optimized JavaScript that automatically works across all major browsers. GWT-Ext is a widget library for GWT that provides rich widgets like Grid with sort, paging, filtering and so on to make our portlet more functional. Google Web Toolkit and GWT-Ext provide the following advantage:

- Enable the developer to create dynamic, AJAX-driven web user interfaces using only Java and take advantage of Java tools like Eclipse that are already available.
- View code changes immediately using GWT's hosted mode browser without compiling to JavaScript or deploying to a server.
- The GWT compiler safely eliminates unused classes, methods, fields, and method parameters.

- Communicate with your server through really simple RPC.
- Since DAS provide standard web service interfaces, the basic communications between the server-side rpc services and DAS web services are based on SOAP over HTTP.

1.2.3 How it works

In the following, the basic interactions between the portlet, DAS and Shibboleth are explained in more detail to clearly depict the operation scenario:

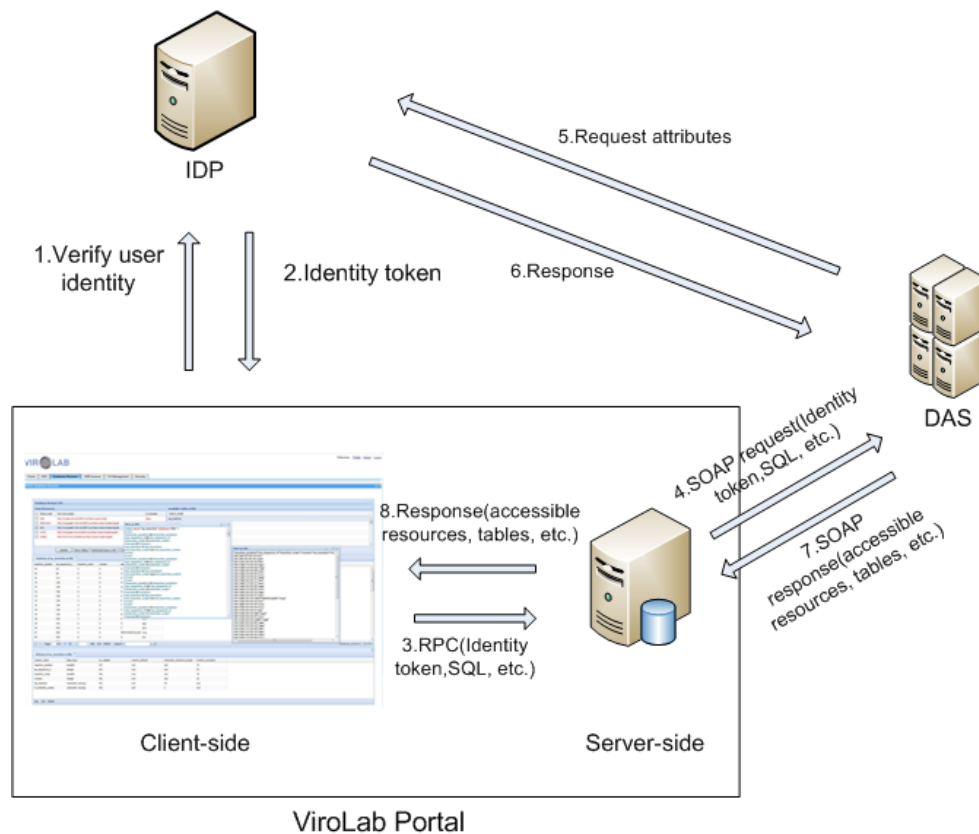


Figure 3: Interactions between Portal, DAS and Shibboleth

1. A ViroLab user uses his credential (usually username and password) to log into the ViroLab Portal via web browser. The credentials are transferred to the identity management system (IDP) of the user's home organization (HO) and verified against the locally stored credentials.
2. In case the user is known to the home organization's user database, a digital identity token (handle) is created and sent back to the portal
3. Once the user is logged in, the client-side Ajax portlet does a RPC call using user handle and IDP address as parameters automatically to authorize the user with the DAS. The user handle, and the IDP address can be obtained from the incoming HTTP POST request (forwarded by the corresponding IDP). This request contains all relevant parameters in form of hidden HTML form fields. If the user is already authorized, different RPCs could be done to browse and query distributed databases through DAS.

4. The server-side RPC services send SOAP requests to DAS to get list of accessible resources, tables, schemas ...
5. DAS uses the received user handle to request the corresponding user attributes like his role, institution or e-mail address from the user's home organization
6. The HO obtains the user attributes from the local database and returns them to the DAS.
7. DAS checks its stored access control policies based on the attributes and returns SOAP response containing a list of accessible resources. If a authorized user want to browse and query distributed databases, DAS will return SOAP responses containing corresponding tables, schemas.
8. Once the server-side RPC services received responses from DAS, the client-side Ajax application will be refreshed with the results.

At Figure 4 the detailed sequence for the interactions between the portlet, DAS and Shibboleth is presented. It is more intuitive and understandable.

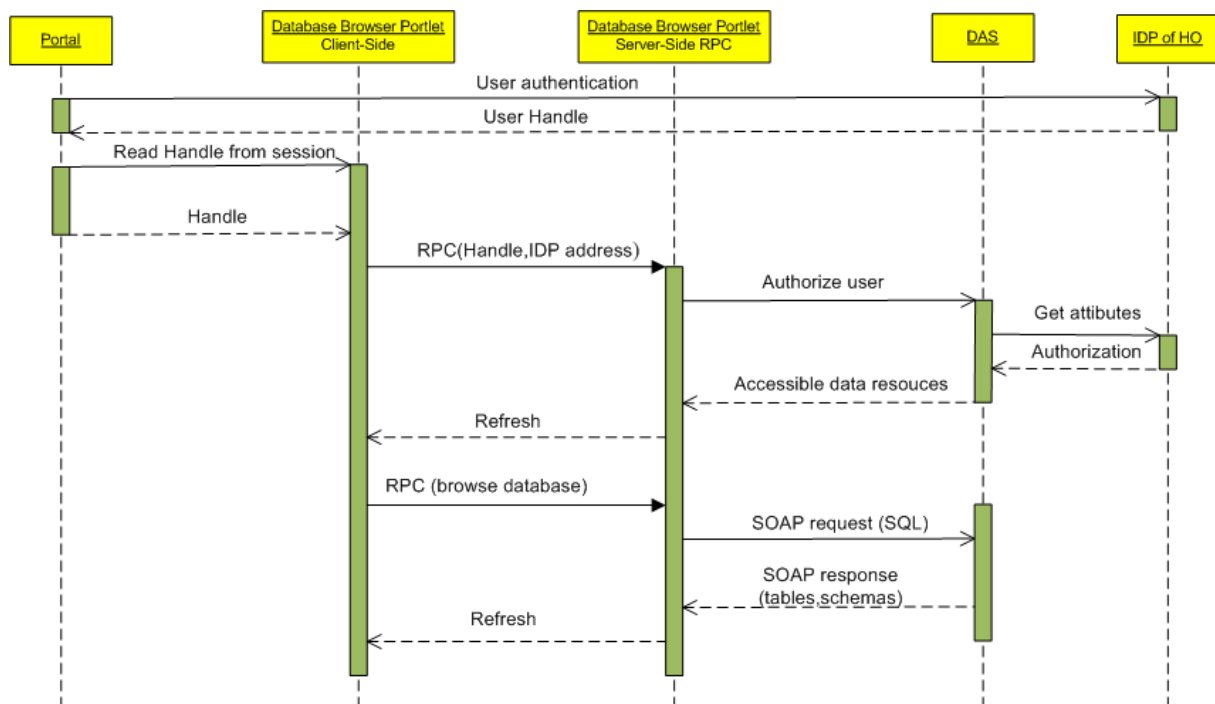


Figure 4: Sequence diagram for the interactions

2 Getting Started

2.1 Requirements

Database Browser Portlet requires a Java Runtime Environment (JRE), version 1.5 or higher. Sun Microsystems provides a JRE available at <http://java.sun.com/j2se/1.5.0/download.jsp>. Ensure that you set up the environment variable JAVA_HOME and add the java binary to your PATH.

Database Browser Portlet requires also installed shibboleth sp and a Tomcat servlet container v5.5 or higher, in which a shibbolized GridSphere is already deployed. The ViroLab edition of shibbolized GridSphere inclusive shibboleth sp is provided by our project partner Gridwisetech (<http://www.gridwisetech.com/>).

2.2 Getting the Files

There are two types of installation options available to you depending on your requirements:

- Database Browser Portlet WAR: This provides the complete Database Browser available as a packaged WAR file that can be dropped into an existing Tomcat servlet container. It is available in ViroLab SVN repository HLRS:
https://svn.gforge.hlr.de/svn/virolab/trunk/modules/DataAccess/Database_Browser_portlet_v0.2/databasebrowser.war.
- Database Browser Portlet Source: This provides the Database Browser source code that can be instantly deployed Tomcat servlet container. It is also available in ViroLab SVN repository HLRS:
https://svn.gforge.hlr.de/svn/virolab/trunk/modules/DataAccess/Database_Browser_portlet_v0.2/databasebrowser

2.3 Installing Database Browser Portlet

2.3.1 Deploy the WAR

The WAR file must be copied into the directory %CATALINA_HOME%\webapps. "%CATALINA_HOME%" is the location where Tomcat was installed. Note that Tomcat expands WAR files in the webapps directory if your tomcat is already running. When updating the WAR file you should delete the corresponding directory which is made by previous versions. Next, restart Tomcat and configure GridSphere to display it (go to Layout in the upper right corner and add it the desired layout), you will see the Database Browser portlet.

2.3.2 Deploy the Source

Building the source code requires that you have Ant installed already. Ant provides a Java based make tool and can be downloaded at <http://ant.apache.org/bindownload.cgi>. Make sure you set up the environment variable ANT_HOME and add the ant binary found in \$ANT_HOME/bin to your PATH. Copy the source folder “databasebrowser” into the Gridsphere source distribution under directory projects (e.g. c:\gridsphere\projects).

Now to compile and deploy the portal into your Tomcat servlet container, simply execute the following from within your gridsphere directory:

```
ant deploy
```

You are now ready to start the portal.

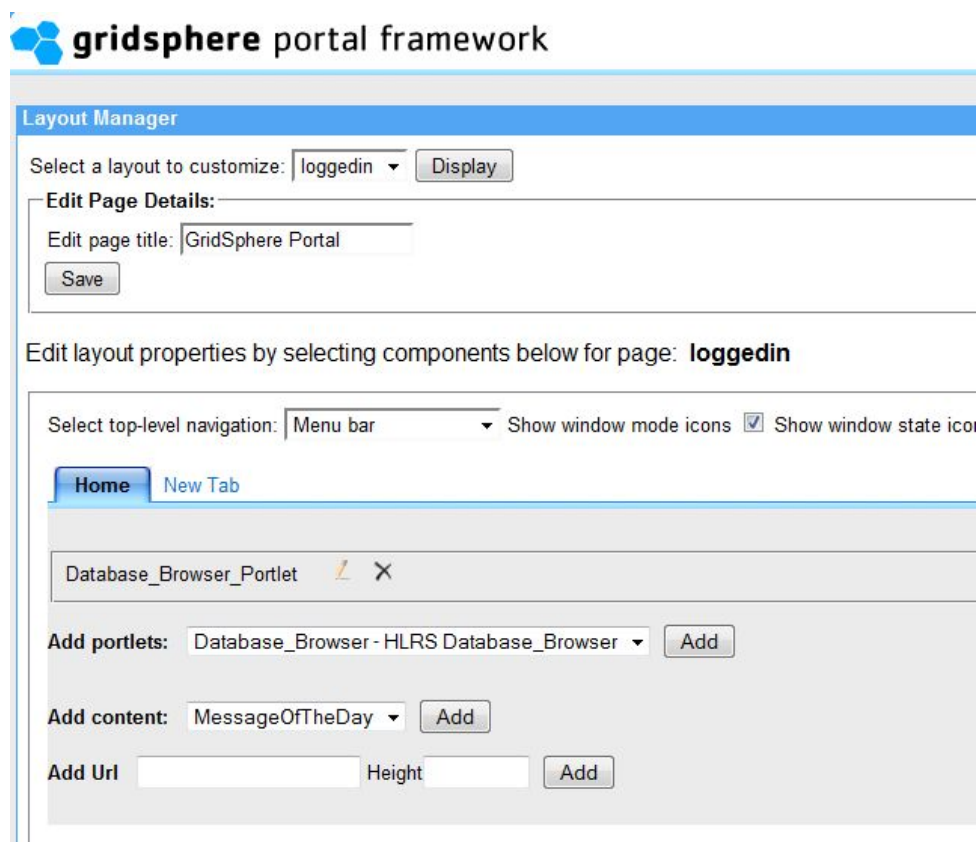


Figure 5 add the portlet in the desired layout

2.4 Operating the Database Browser

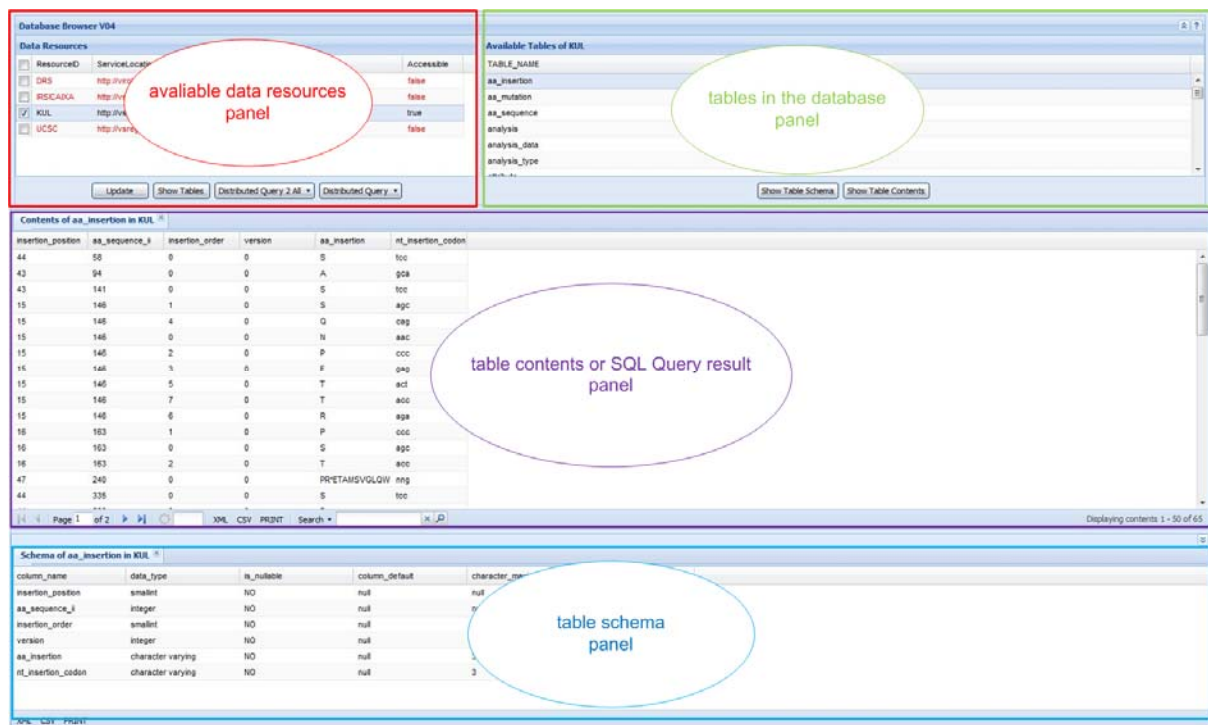



Figure 6 Database Browser GUI

The database browser has four panels for displaying different information and interacting with Users. The north panel (available data resources panel and table panel) and south panel can be hidden by click the  tool in order to have a full screen view of table contents.

If a user has successfully logged in to the portal via the WAYF (“Where Are You From”) module and selects the Database Browser Portlet application, he/she directly obtains an overview of all available resources displaying in the “available data resources panel”. As mentioned above the Database Browser did already authorize the user with the user identity (Handle) and IDP address from session inside portal during the loading. Non-accessible resources are shown in red color.

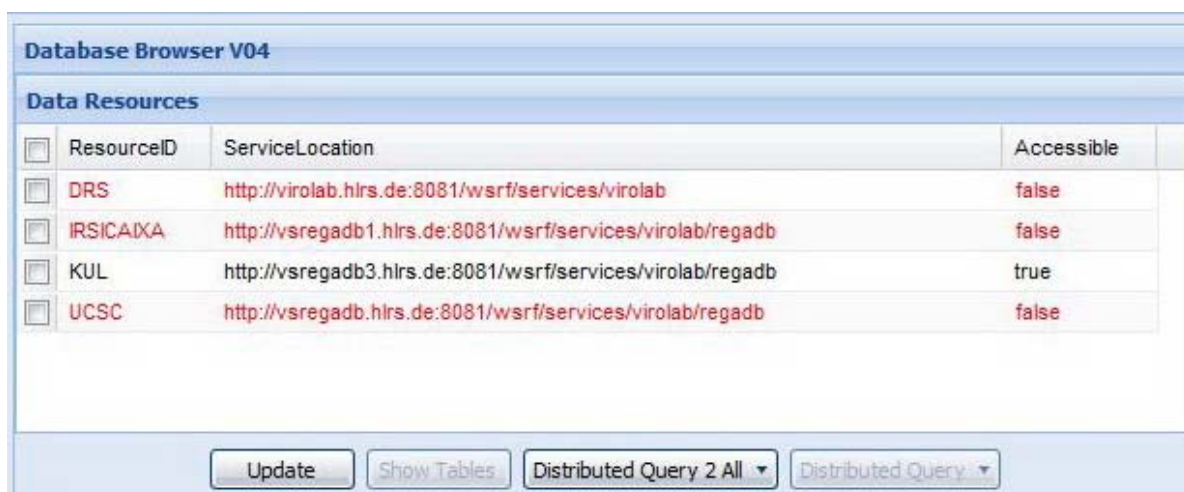


Figure 7 Data Resources Panel

2.4.1 Browsing accessible database

All listed accessible resources are connected via the DAS, and can be used for further processing. If you want to browse the tables in data resource:

1. Just select the corresponding resource then click the **Show Tables** button or double click on the corresponding data resource.
2. The tables in the database will be displayed in the “table panel” on the right side.

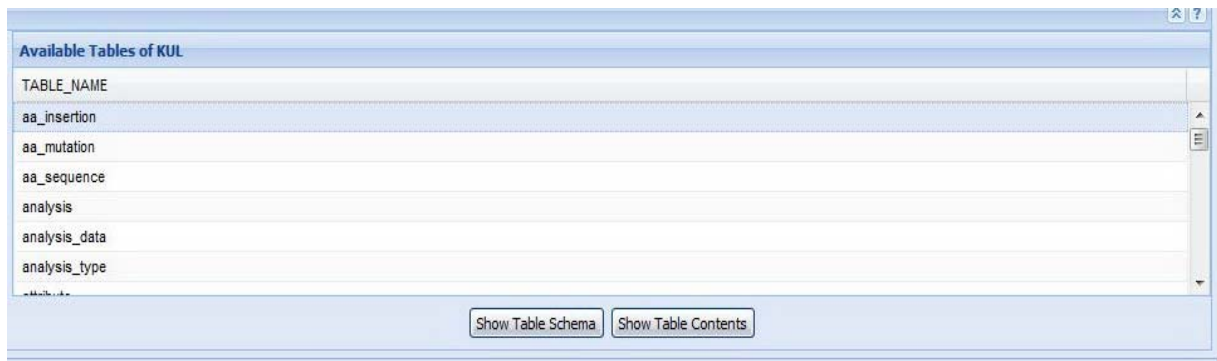


Figure 8 Table Panel

2.4.2 Getting the table schema

1. Select the corresponding table in the “table panel”
2. Click the **Show Table Schema** button.
3. Then the table schema will be displayed as a new tab in the bottom “table schema panel”.

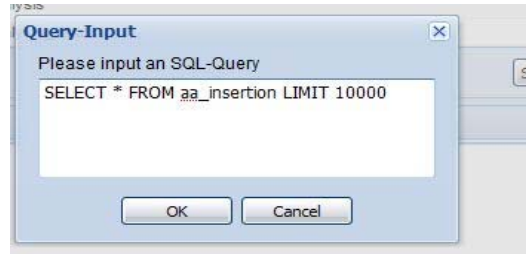
Schema of aa_insertion in KUL					
column_name	data_type	is_nullable	column_default	character_maximum_length	numeric_precision
insertion_position	smallint	NO	null	null	16
aa_sequence_ii	integer	NO	null	null	32
insertion_order	smallint	NO	null	null	16
version	integer	NO	null	null	32
aa_insertion	character varying	NO	null	30	null
nt_insertion_codon	character varying	NO	null	3	null

XML CSV PRINT

Figure 9 Table Schema Panel

2.4.3 Getting the table contents

1. Select the corresponding table in the “table panel”
2. Click the **Show Table Contents** button.
3. After the “Show Table Contents” button is clicked, the following dialog should come and all others should be masked.



Because some tables in our database are really huge, so it is necessary to list the table contents with a limitation, by default the limitation is 10000.

4. Modify the limitation if you want, then click the “OK” button to execute the SQL query. Otherwise click the “Cancel” button to return to the database browser.
5. The table contents will be displayed as a new tab in the central “table contents panel”.

Contents of aa_insertion in KUL					
insertion_position	aa_sequence_id	insertion_order	version	aa_insertion	nt_insertion_codon
44	58	0	0	S	tcc
43	94	0	0	A	gca
43	141	0	0	S	tcc
15	146	1	0	S	agc
15	146	4	0	Q	cag
15	146	0	0	N	aac
15	146	2	0	P	ccc
15	146	3	0	E	gag
15	146	5	0	T	act
15	146	7	0	T	acc
15	146	6	0	R	aga
16	163	1	0	P	ccc
16	163	0	0	S	agc
16	163	2	0	T	acc
47	240	0	0	PR*ETAMSVGLQW	nng
44	335	0	0	S	tcc

Page 1 of 2 XML CSV PRINT Search

Figure 10 Table Contents Panel

P.S. if you do double click on the table name in “table panel”, a new tab of table schema and a new tab of table contents will be respectively displayed in corresponding panel.

2.4.4 Handling the data

All tabs in both panels are closeable. Due to the fact that too many tabs will make the database browser sluggish, so please close the tab if you don't need it anymore.

In each tab, you can click the table head to sort rows ascending or descending.

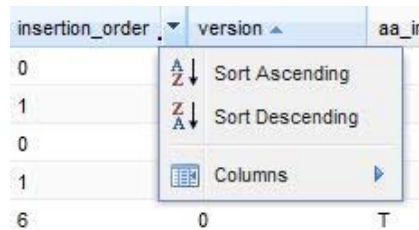


Figure 11 Sorting the Table

Click the downward arrow that appears on the right side of table header, you can find menus for configuring the sorting.

There are three buttons at the bottom of each tab in both panels:

- Click the **XML** button, the data displayed in the tab will be shown as XML in a detached window in order that you can print the data as XML or save the XML to your local storage.

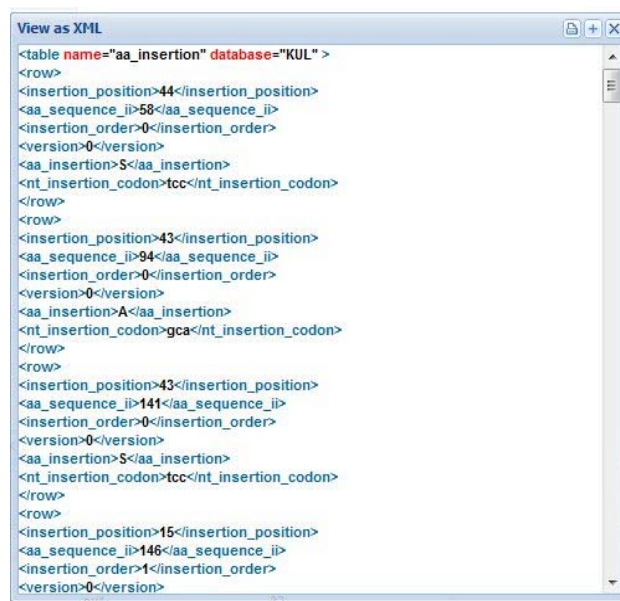


Figure 12 XML Window

At the top right corner of the new window, there are also three tool buttons.



: print the XML file



: copy the XML data to Clipboard (it works only under Microsoft windows)



: close the window

- Click the **CSV** button, the data displayed in the tab will be shown in CSV-format in a detached window in order that you can print the data as CSV or save the CSV file to your local storage. The window has the same tool buttons as described above.

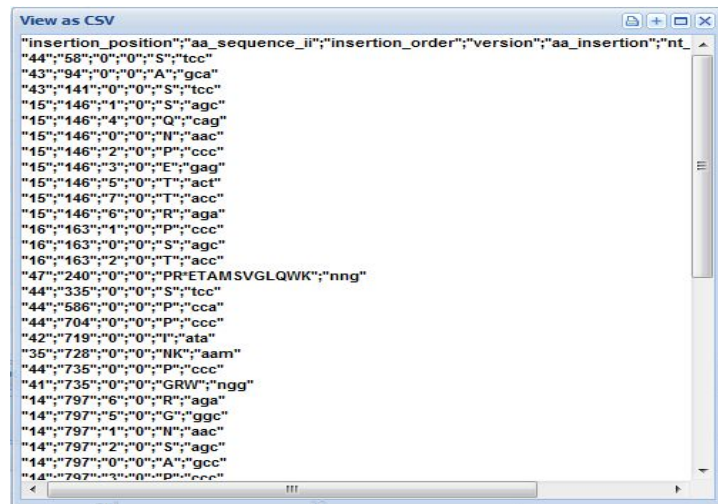
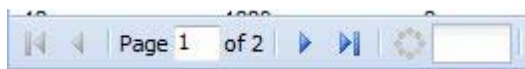


Figure 13 CSV Window

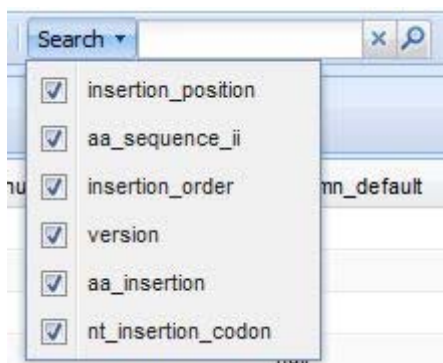
- Click the **PRINT** button, the data displayed in the tab will be printed as a HTML-table.

Because the table contents are usually more abundant and useful than table schema for the users, therefore you can find more tools on the bottom toolbar in the central “table contents panel” for facilitating user interaction:

- Table contents are displayed as grid with paging due to its large number of rows, and by default each page shows 50 rows. Through the toolbar you can browse the pages and define the number of rows that you want shown in each page.

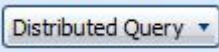


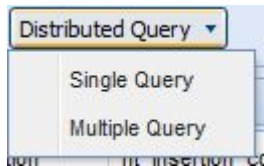
- You can also search rows whose values exactly match your keyword input.



2.5 Executing Distributed SQL queries

Having selected the corresponding resource, you can also send immediately one or multiple SQL queries to one or multiple data resources concurrently.

1. Click the  button, the following submenu will arise and you can select the corresponding submenu to execute single or multiple SQL queries.



2. If “Single Query” submenu is clicked, the following “DAS Query Tool” will arise. DAS provides an additional database for each user to store their SQL queries. The “DAS Query Tool” will connect the database and get stored queries during initiation. You can select the queries that you did already stored from the drop-down menu or input a new query in text field, then click “OK” button to execute it. Once a query is successfully executed, the “Store” button will be enabled and now you can store it for future use.

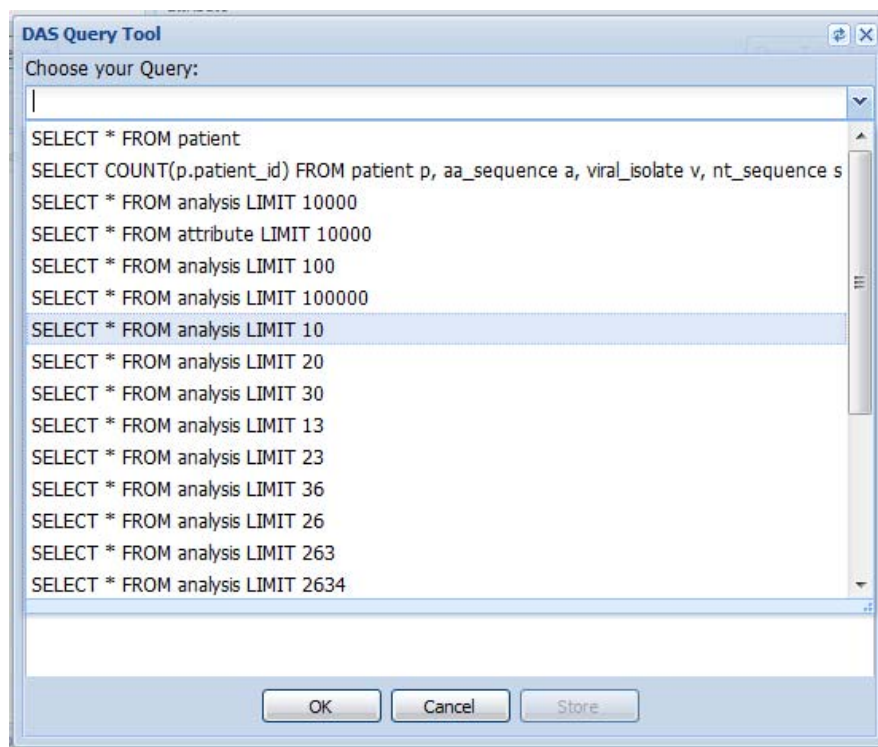


Figure 14 DAS Query Tool for Single Query

3. If “Multiple Query” submenu is clicked, the following “DAS Query Tool” will arise.

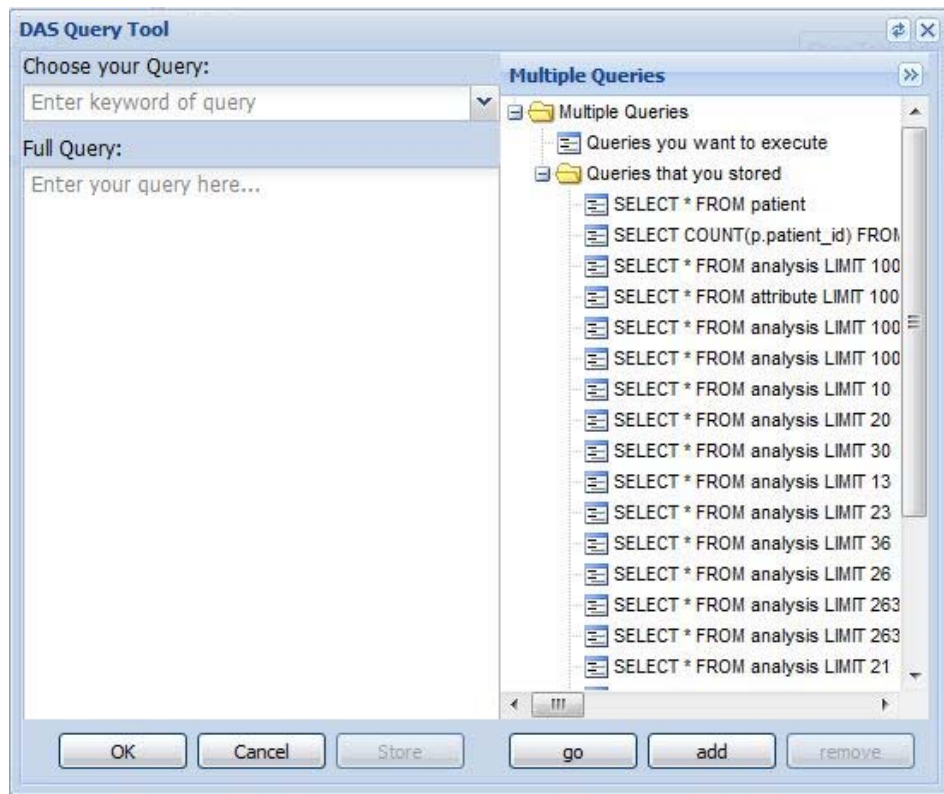


Figure 15 DAS Query Tool for Multiple Queries

It has an additional tree panel on the right side. All queries under the tree branching “Queries you want to execute” will be executed. There are four ways to build a multiple-queries-tree:

- Choose a query from the drop-down menu then the query will be displayed in the field “Full Query”. Next, click “add” button then the selected query is added to the tree branching “Queries you want to execute” as child node.
- Click on a child node under “Queries that you stored” then the full query will also be displayed in field “Full Query”. Next, click “add” button to add the selected query to the tree branching “Queries you want to execute” as child node.
- Or just input your new query in the field “Full Query” then click the “add” button.
- Click on a child node under “Queries that you stored”, drag and drop it to node “Queries you want to execute”.

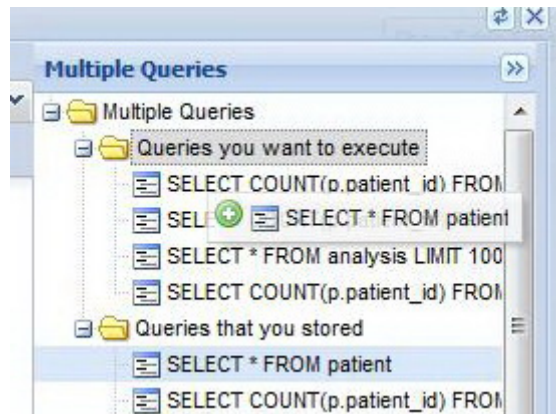


Figure 16 Building the Multiple-Queries-Tree

After the multiple-queries-tree is constructed, you can click the “go” button then the queries will be executed and outcome of each query will be displayed as a new tab in the central panel.

4. Click the Distributed Query 2 All ▼ button, then you can execute single query or multiple distributed queries to all accessible data resources via the “DAS Query Tool”.

3 References

- [DA07] M. Assel, B. Krammer, and A. Loehden. Data Access and Virtualization within ViroLab. In Proceedings of the 7th Cracow Grid Workshop 2007, pp. 77-84, Kraków, Poland, October 16-18, 2007. http://www.virolab.org/documents-repository/doc_details/33-data-access-and-virtualization-within-virolab.html
- [D2.3] Deliverable 2.3 ViroLab VO version 1 deployment, integration with WP3, WP4 and WP5 - report and demonstration. http://www.virolab.org/documents-repository/doc_details/83-d23-virolab-virtual-organisation.html
- [GS] GridSphere Project. <http://www.gridsphere.org>
- [SBL] Shibboleth Project. <http://shibboleth.internet2.edu/>
- [JSR] JSR 168: Portlet Specification. <http://www.jcp.org/en/jsr/detail?id=168>
- [GWT] Google Web Toolkit Project developed by Google, licensed under the Apache License, v. 2.0. <http://code.google.com/webtoolkit/>
- [GWTE] GWT-Exe Project, licensed under the GNU Lesser General Public License (LGPL), v 3.0. <http://gwt-ext.com/>

Filter Table Content feature

By this new feature you can retrieve the records needed from the table without need to write the SQL query.

Click the **Filter Table Contents** button to filter the selected table (patient) as in figure 1 below.

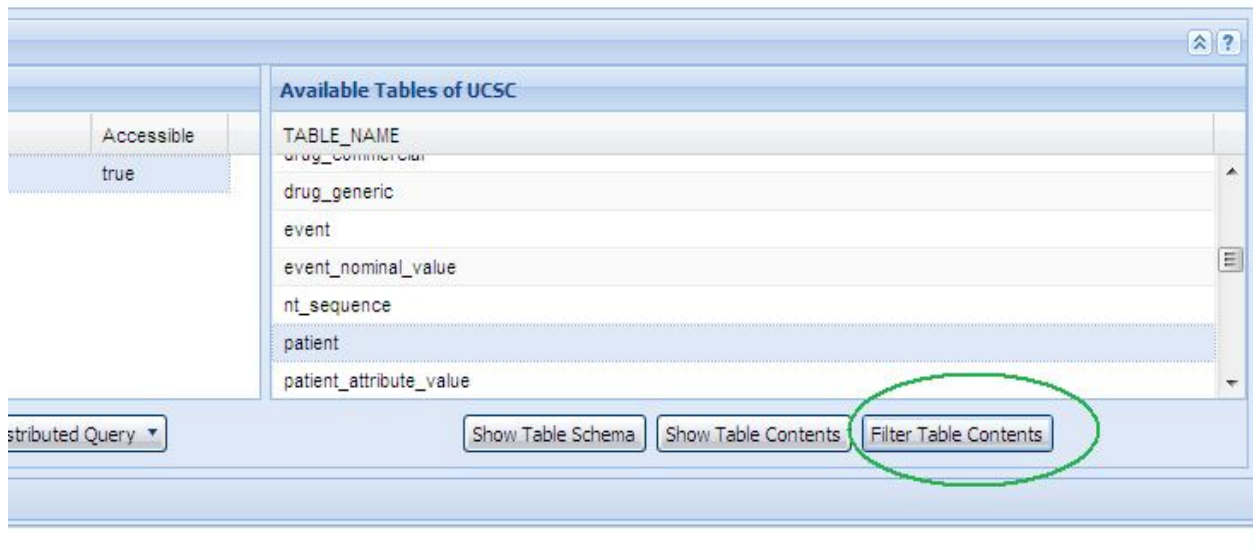


Figure 1

As a result a new table is listed as in figure 2 below

Table Schema Filter Results of patient ✕								
Limit of Records:		4130	Get Results		Check SQL query: <input type="text"/>			
Column Name	Select Column	Type	Max	From	To	From Date	To Date	Like
patient_id	<input type="checkbox"/>	integer	4130	0	4130			
version	<input type="checkbox"/>	integer	0	0	0			
patient_id	<input type="checkbox"/>	character varying						
last_name	<input type="checkbox"/>	character varying						
first_name	<input type="checkbox"/>	character varying						
birth_date	<input type="checkbox"/>	date	2029-10-16				2029-10-16	
death_date	<input type="checkbox"/>	date	2008-07-26				2008-07-26	

Figure 2

The table shown in figure 2 is used to prepare the SQL statement to retrieve records from the patient table.

The first column contains the name of the columns of the patient table, you can select the names of the columns you want to retrieve by check the box in the second column (**Select Column**), the **Type** column contains the data type of each column in the patient table, the **Max** column contains the maximum value of the integer, small integer and date types of the patient table.

From and **To** columns belong to the integer and small integer types, they are initialized to zero (**From**) and the maximum value (**TO**), you can insert the range you want to be used as a condition for the where statement in the SQL query.

From Date and **To Date** column belong to the Date data type, you can insert the range you want for the SQL where condition.

Like column belongs to the character varying and text data types, you can insert the condition for the string selection here.

You can't update any cell unless the column name (**From, To, From Date**etc) and the data type (integer, small integer, date,....etc) are compatible as mentioned in the previous three paragraph, e.g **From** and **To** column are just belong to integer and small integer data type, so you can't insert anything in cells of these column if the data type of this cell is date, text or character varying.

Check SQL query button used to check the SQL statement before execute it, the SQL statement is shown in the text area to the right of the button, if the statement colour is **RED**; thats mean you have an error in your selection, so recheck it again; otherwise the colour is black.

Limit of records text box initialized to the maximum number of records in the selected table and you can use it to limit the number of records you want to retrieve from the that table.