Astronomical Data Analysis Software and Systems XX
ASP Conference Series, Vol. 442
Ian N. Evans, Alberto Accomazzi, Douglas J. Mink, and Arnold H. Rots, eds.

VODance: VO Data Access Layer Service Creation Made Easy

Riccardo Smareglia,* Omar Laurino, and Cristina Knapic

INAF — Astronomical Observatory of Trieste

**smareglia@oats.inaf.it*

Abstract. We present a tool for rapid deployment of Virtual Observatory compliant services. Users who want to publish their datasets to the Virtual Observatory can achieve this goal without having to deal with the technical details of standard services development and without having to move their data. With VODance users just have to provide a database connection to our center that points to their available data and fill out a metadata description form without having to export their data. Data Access Layer services are created on the fly and published, through the Italian Astronomical Archive Center (IA2), to the Virtual Observatory. VODance has been successfully used to publish Cone Search and Simple Image Access Protocol services out of MySQL and Oracle database management systems.

1. Introduction

The International Virtual Observatory Alliance (IVOA) provides standard protocols and formats for sharing astronomical information. Thus, in order to publish data to the Virtual Observatory community, data providers are to implement web services which are compliant to the Data Access Layer (hereinafter DAL) IVOA standard specifications.

Some of these protocols are widely used by the most popular VO clients, e.g., Topcat, Aladin, and many Virtual Astronomical Observatory web applications.

With VODance we can dynamically create compliant DAL services on the fly at runtime, out of a generic database table or view, either local or remote.

2. VODance Features

2.1. Requirements

VODance allows our data center to create on the fly VO compliant DAL services out of a database table or view, either on a local or a remote database.

The basic requirement for VODance was the ability to create an arbitrary number of DAL services at runtime by simply filling in a metadata description of the service and of the individual columns. The data to export are supposed to be present in the form of a database table or view. The only requirement for the database is a JDBC Java driver to be available for the connection.

The basic functional requirement for which VODance has been created is depicted in Figure 1.

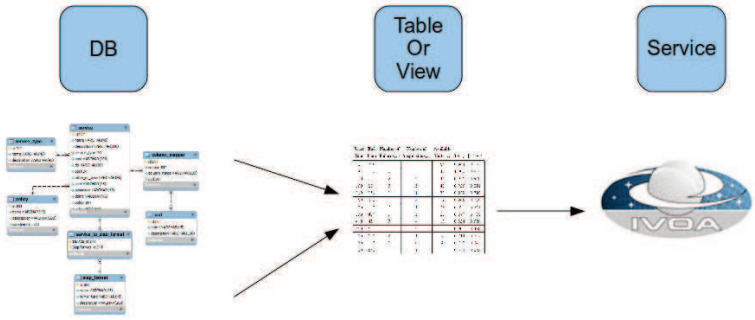

Figure 1. VODance allows the creation of compliant DAL services out of a database table or view, either on a local or remote database, on the fly at runtime.

2.2. Service Creation Form

In order to create a new service users can use the administration web application, by means of a simple form. A connection to the database table or view to be exported has to be available on the server on which VODance runs, which is not necessarily the server on which the administration web application runs.

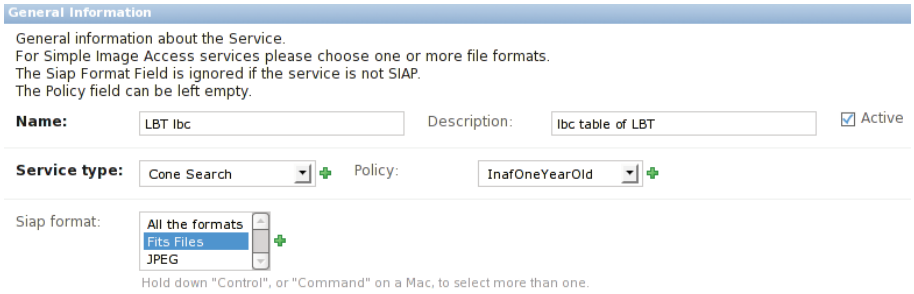

Figure 2. Service generic information.

The procedure by which a new service is created follows these steps (see Figure 2 and Figure 3):

1. Define general parameters,
2. Define a data access policy, if necessary,
3. Fill in database connection parameters,
4. Define column metadata,
5. Save the newly created service.

As soon as the form is submitted, the new service is up and queryable. The only extra step needed is to register the service in a VO resource registry in order for it to be indexed and discoverable.

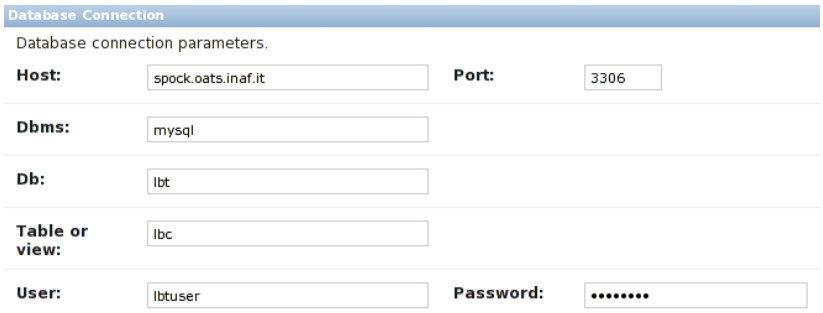

Figure 3. Database connection information.

3. Data Access Policy

In some cases, it may be necessary to filter out some rows from the results of the query; for example, if the table contains proprietary data or data from different partners in a collaboration, some records have to be filtered out.

In order to achieve this, one might create a new view on the database. However, VODance offers the capability to create dynamic data access policies, so that no changes are needed in the database.

Policies can be created, combined, added, and removed dynamically at runtime, on the fly, and without touching the database.

4. Column Metadata

Figure 4. Column metadata descriptions. Columns can be aggregated into arrays and dynamic type casting can be applied to column data.

When defining column metadata, VODance allows users to set the UCD and units strings for each column. The UCD definition is particularly important for mandatory fields which are used for the query. In particular, by defining the right ascension and declination columns, VODance knows which fields have to be constrained in the query to the specific table. For these columns, moreover, units strings are useful to inform VODance that a conversion between the standard query input (column degrees) and the actual column values (which might be radians, for example) is required.

The column mapper interface also provides two advanced features: aggregate columns and dynamic type casting.

Aggregate columns allow the definition of a new column that contains arrays of values from different columns in the original table. For example, the SIAP protocol

dictates that one column must contain an array with the number of pixels of both the *x* and *y* image axis. A specific syntax in the form `column1, column2 = column3` informs VODance that values stored in `column1` and `column2` must be inserted in the new `column3` as arrays.

Dynamic type casting is necessary in order to be sure that table columns meet the requirements of the standard DAL service. For example, if a column stores values as floating point numbers while the service specification dictates that the information must be an integer, the casting can be made at runtime. Besides, some JDBC drivers and DBMS specific issues may arise from the query, so that values can be presented to VODance as Java objects that are not easy, or impossible, to serialize to XML.

For example, all the columns created as `NUMBER` in Oracle databases are apparently rendered as `java.math.BigDecimal` objects by the JDBC driver.

Dynamic type casting allows fine grained control over these issues, by allowing the definition of specific and reusable TypeCast entities.

5. Tested Services

We are currently publishing several Cone Search (quasar photometric candidates in the SDSS, photometric redshifts for SDSS galaxies and quasar photometric candidates) and SIAP services (TNG) using two different DBMS's: Oracle and MySQL.

In principle, all DBMS's for which a suitable JDBC driver is available can be used as a backend for retrieving data. However, issues may arise from the specific implementations, in particular for the SQL query syntax.

6. Deployment Environment

VODance is currently deployed in a cluster environment. In particular, a number of shared-nothing instances of application servers host the web application. The internal database is in turn hosted on an high available MySQL cluster, and software load balancing is performed by means of Linux Virtual Server, in a redundant configuration.

A file server web application is deployed in the shared-nothing instances as well, for serving images according to the SIAP services.

The administration user interface is deployed on a different web server that shares the internal database cluster with the web application that actually serves the DAL services.

7. Distribution

Since the architecture depicted in § 6 is quite elaborate, consisting of a stack of different services and servers, VODance will be distributed as a virtual application, i.e. as a disk image intended to be run in a virtual machine.

With this choice we hope to address several issues: first of all, final users will not be required to install package and configurations, except for the unavoidable network configuration; secondly, multiple instances of the application can be combined allowing us to achieve with minimal effort the high level of availability and redundancy that is typical of a production environment.