JVO Query Language

Naoki Yasuda

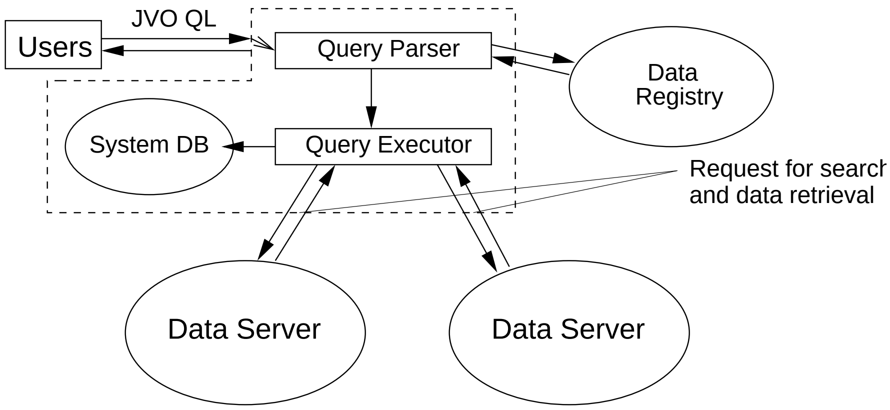
February 28, 2003

JVO will provide the function to retrieve catalog data and image data from multiple data servers via single user interface. Here we will discuss the grammar of JVO Query Language which describes the request for data retrieval.

1 Database configuration

In JVO, the following database configuration is assumed. There are several remote data servers of catalog and/or image data, and JVO will request them for searching and retrieving data. JVO system will have its own database system which will store the data retrieved from remote data servers. In most cases, real materialized data will be stored into JVO own database since these data will be served for the input of further analyses done at JVO system.

In the data registry, basic information about catalog data and image data which remote data servers hold are stored; host names to access, methods to access, metadata of data, and so on. Data registry will be something like UDDI in Web Services.



2 Specification of JVO Query Language

We designed JVOQL as an extension of SQL since SQL has simple and well defined grammar and many people are accustomed with SQL.

In the following, `catalog` means catalog data in tabular format and `data` means image data.

JVO specific operands which are extension to SQL are written in capital letters.

Queries in JVO can be categorized into the following 3 category.

(a) Queries which deal with only catalogs.

(b) Queries which deal with catalogs and images. But images are pointed by the columns in catalogs.

(c) Queries which deal with images without catalog.

○ In the case of (a) and (b)

```
create [materialized] view mytable as
select c1.column1,
       c1.column2,
       c2.column1,
       c2.column2,
       ...,
       d1.BOX(POINT(c1.ra, c1.dec), width1, height1) as BlueImage,
       ...,
       c1.column3 / c2.column5 as flux_ratio,
       c3.*,
       ...
from   catalog1 c1,
       catalog2 c2,
       catalog3 c3,
       ...,
       data d1,
       ...
where  XMATCH(c1, c2, !c3, ...) < 3 arcsec
       and
       (c1.column1 - c2.column1) < 6.0 mag
       and
       BOX(POINT(ra0, dec0), width0, height0)
       ...
```

create clause

`mytable` specifies the name of new table created in the internal database. Column names in `mytable` will be determined according to the specification in `select` clause. Metadata such as data type will be inherited from the metadata of original catalogs.

We are assuming that all catalogs both on external data server and on internal database are accompanied with metadata such as names of columns, data types, UCDs, and units. Catalogs in JVO can be considered as mapping of VOTable into relational database.

When `materialized` is specified, queries to data servers are actually executed and the results will be stored in the internal database.

select clause

```
column : [catalog.]UCD [as name] |
         [catalog.]column_name [as name]
```

`column` specifies the column of catalogs on data server.

If a user knows the catalog specific column names, user can use that name (`column_name`) to issue queries. If not, user can use UCDs. For example, magnitude in B band is `PHOT_MAG_B`.

When `as name` is specified, that column will be stored in `mytable` using `name` as a column name. If not specified, column name will be the original name (`catalog1.column1`) such as `TWOMASS.ra`.

```
data : [data.]AREA(...) [as name]
```

`data` specifies the area for the image data which will be cut out from data servers. Each image data corresponds to each record in a catalog. Detailed format of `AREA` will be given later, but in general case area will be rectangle and expressed as

`BOX(POINT(c1.ra, c1.dec), width1, height1)`.

This means cutting out rectangular area of width `width1` and height `height1` whose center position is determined from the values of column `ra` and column `dec` of catalog `c1`.

On real tables of database system, the values of columns corresponding to the image data will be a link to the data. This link can be file names on the local file system, URL to the remote data servers, or the image cutout request to the remote data servers.

All columns in a catalog can be specified as `c3.*`. In this case, JVO system will get metadata from data server and expand to each column names.

Arithmetics among columns can be specified.

from clause

```
catalog : KEYWORD1 & KEYWORD2 ...
                  [select [by [MAX|MIN](PROPERTY) | ALL]] [name] |
          [server.]catalog_name
                  [select [by [MAX|MIN](PROPERTY) | ALL]] [name]
```

`catalog` specifies the name of catalogs stored in the remote data servers and internal database. Same as for `column`, if individual names are known, those catalog names (catalog_name) can be specified. If the name of data servers are known, that can be added prior to the catalog name.

When individual catalog names are not known, keywords which describe the target catalog can be specified. KEYWORDs are the words to characterize the catalog like the content of catalog such as "cosmic ray", "SiO masar", or telescope/instrument name which data are taken. Multiple KEYWORDs can be combined for more specific selection of catalog.

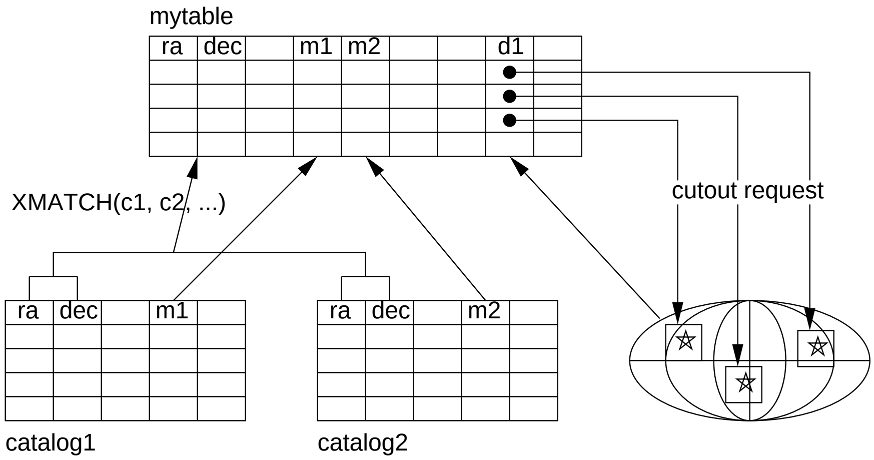
When KEYWORDs are specified, JVO system will look up corresponding catalogs from data registry. When multiple catalogs are found, operands like select by PROPERTY and ALL will select among them. PROPERTY is the index which will rank catalogs, such as "detection limit", "measurement accuracy", and they are also stored in the data registry. ALL means all selected catalogs are used for search.

When ALL is specified, the results of query from each catalog will be concatenated into one table in the direction of inserting records.

name will specify the alternate name of the catalog in this query.

```
data : KEYWORD1 & KEYWORD2 ...
             [select [by [MAX|MIN](PROPERTY) | ALL]] [name] |
      [server.]catalog_name
             [select [by [MAX|MIN](PROPERTY) | ALL]] [name]
```

data will specify the name of image data stored in the remote data server. The way for specification is same as catalog. Note that ALL is specified for image data, all image data will be searched for each record of catalog, which will be added into table as new columns.



where clause

XMATCH operand means cross matching multiple catalogs.

```
XMATCH(c1, c2, !c3, ...) < 3 arcsec [NEAREST | BRIGHTEST | ALL]
```

means that cross matching using positional information between `c1` and `c2` with an error of 3 arcsec and select objects which are not included in `c3`.

When multiple objects are matched for one object within a specified error, `NEAREST`, `BRIGHTEST`, and `ALL` will specify how to select objects among them. If `ALL` is specified, all the combination will be listed up.

Though `(c1.column1 - c2.column1) < 6.0 mag` looks like the conditional statement in the normal SQL language, unit conversion among catalogs will be done using the information in the metadata of the catalogs before comparison.

- Area specification of celestial sphere

```
AREA : [INSIDE | OUTSIDE] AREA0 |
       AREA1 [OVERLAP | UNION] AREA2 |
       SHAPE

SHAPE : BOX(POINT(ra, dec), width, height[, Position Angle]) |
        CIRCLE(POINT(ra, dec), radius) |
        OVAL(POINT(ra, dec), radius1, radius2,[, Position Angle]) |
        TRIANGLE(POINT(ra1, dec1), POINT(ra2, dec2), POINT(ra3, dec3))
```

`POINT(ra, dec)` specifies a point on the sky. If `ra` and `dec` are columns in a catalog, the coordinate (equatorial, galactic, or something else) of them will be judged from their UCDs.

`INSIDE`/`OUTSIDE` means inside/outside of the area, respectively. Specification without both of them works as `INSIDE` is specified.

`AREA` can be defined as overlap (`OVERLAP`) and/or union (`UNION`) of multiple `AREA`s. Each `AREA` can be a rectangle (`BOX`), a circle (`CIRCLE`), an oval (`OVAL`), or a triangle (`TRIANGLE`). For `BOX` and `OVAL`, position angle can be specified. If not specified, axes will be aligned with equatorial coordinates.

With these definitions, arbitrary shapes on the celestial sphere can be expressed to some extent.

- Time range specification

Specify by starting time and range. Union of these can be allowed.

- Wavelength range specification

Specify by general filter names, whether specified wavelength is within the bandwidth of each data, or whether central wavelength of each data is within specified wavelength range. Each filters' sensitivity is assumed to be stored in the data registry.

Operation of tables

To create new table from multiple tables (for example, merge tables for independent areas into one table), the following commands can be used. All the tables have to have same configuration on columns.

```
create view newtable as
       select * from table1
       where [not] exist table2

create view newtable as
       select * from table1
       union select * from table2
```

The former command will select records common to both `table1` and `table2`. If `not` is specified, select records in `table1` but not in `table2`. The latter command will concatenate two table.

Cutout of table

To create compact table which contains the records with some characteristics and interesting columns from big table, use the following command same as creating view in SQL.

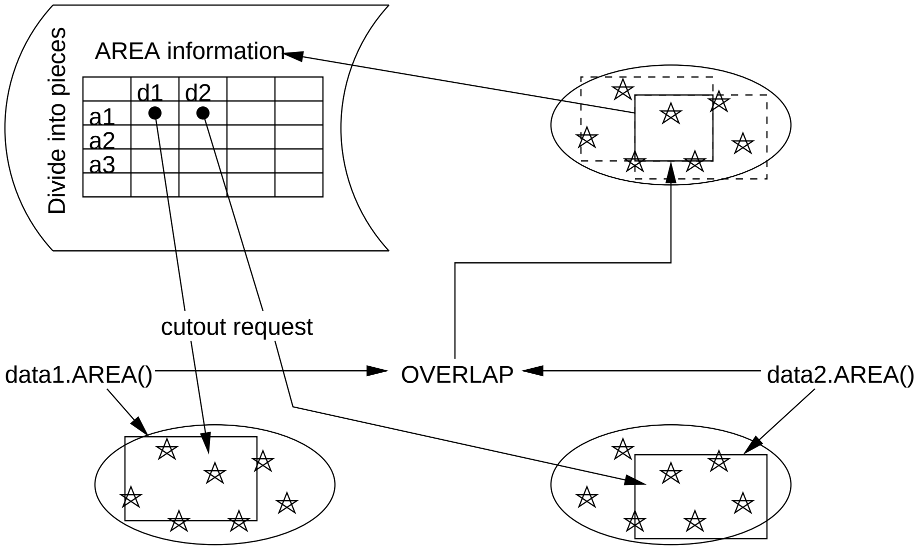
```
create view newtable
       as select column1, column2, ...
       from table
       where column3 = obj_star and
             ...
```

○ In the case of (c)

In this case, what we have in mind is search for the data of area where observed in different wavelength and search for data of area where observed at different time.

As a general formula, the following syntax can be used.

```
select d1.a,
       d2.a,
       ...
from  [server1].data1 d1,
      [server2].data2 d2,
      ...
where AREA as a
```



AREA specifies the area search for the data. If it can be expressed by AREA operands described before, that expression will be used. If not, user defined functions will be used. One suggestion for the interface of user defined functions is that output will be a list of HTM (Hierarchical Triangle Mesh) indices. For the data of time series, this list will be a pair of HTM index and time.

The retrieved data will be constructed from many data files. However, the retrieved data will be stored in the system as if that is one set of data for users and the same image access method can be applied same as for the external data servers.

The followings are simple examples.

- Retrieve the data for the area where 2 colors are available

```
select X.a,
       Y.a
from   data1.wavelength1 X,
       data2.wavelength2 Y
where  (X.AREA() OVERLAP Y.AREA()) as a
```

`X.AREA() = data1.wavelength1.AREA()` represents the area where the data in an image data archive `data1` covers in a wavelength `wavelength1`.

- Retrieve the data for the area where observed at 2 different times

```
select X.a,
       Y.a
from   data X, Y
where  (DIFF(X.obs_date, Y.obs_date) > 30 days)
        and
        X.AREA() OVERLAP Y.AREA()) as a
```