Astronomical Data Analysis Software and Systems XIV **P1-1-23**
ASP Conference Series, Vol. XXX, 2005
P. L. Shopbell, M. C. Britton, and R. Ebert, eds.

Structured Query Language for Virtual Observatory

Yuji Shirasaki, Masatoshi Ohishi, Yoshihiko Mizumoto, Masahiro Tanaka, Satoshi Honda, Masafumi Oe

National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka City, Tokyo 181-8588, Japan

Naoki Yasuda

Institute for Cosmic Ray Research, University of Tokyo, 5-1-5 Kashiwa-no-Ha, Kashiwa City, Chiba 277-8582, Japan

Yoshifumi Masunaga

Ochanomizu University, 2-1-1 0tsuka,Bunkyo-Ku, Tokyo 112-8610, Japan

Abstract. Currently two query languages are defined as standards for the Virtual Observatory (VO). Astronomical Data Query Language (ADQL) is used for catalog data and Simple Image Access Protocol (SIAP) is for image data. As a result, when we query each data service, we must know in advance which language is supported on the service and then need to construct a query language accordingly. The construct of SIAP is simple, but they have a limited capability. For example, there is no way to specify multiple regions in one query, and it is difficult to specify complex query conditions. In this paper, we propose a unified query language for any kind of astronomical database on the basis of SQL99. SQL is a query language optimized for a table data, so to apply the SQL to the image and spectrum data, the data structure need to be mapped to a table like structure. We present specification of this query language and an example of the architecture for the database system.

1. Introduction

At present, two kinds of data query language are defined as standards in the Virtual Observatory. One is the parameter query, which is used for image data search and called as "Simple Image Access Protocol" or SIAP. Search criteria are specified by a set of "key" and "value" pairs. Another one is a structured query language, which is used for catalog data query and called as "Astronomical Data Query Language" or ADQL (Yasuda et al. 2004). ADQL can specify more complex search criteria than the SIAP do, and also has ability to join multiple tables and can select values derived from the DB columns. So it is worthwhile to adapt the ADQL to perform an image query. The high flexibility of the ADQL syntax, however, causes difficulties to develop an ADQL compliant data service. It is known that incompatibility among the various DBMS products is due to the complexity of the SQL specification. The success of the Virtual Observatory project depends on uniformity of the interface of all the astronomical data services, so we must avoid the complexity and make it simple and easy to implement in defining the standard query language.

2. Syntax Specification

In order to realize the interoperability among the distributed data archives, we need to define a standard query language. As the standard query language must be supported by all the data services, it's specification should be simple and clear for easy implementation. On the other hand, some data service require more sophisticated query syntax to allow users to specify more efficient search criteria. Thus we defined basic syntax which must be supported by all the data services and allowed several enhancements on the basic syntax.

2.1. Basic syntax

The following restrictions are applied to the "select" command of SQL99 specification.

1. Only column name or "*" is specified in the selection list, so an algebraic expression and a function are not allowed. So the role of the "select" clause is just to specify the columns you want to get. The calculation of the derivative from those columns should be done on a portal server or by users themselves.
2. Only one table is specified in the "from" clause, so any type of table join syntax is not allowed. The join of the tables should be done on a portal server or by users themselves.
3. Allowed comparison operators and predicates are `=`, `<`, `>`, `>=`, `<=`, `<>`, `within`, `contains`, `overlaps`, `LIKE` and `BETWEEN`, and any other operators are not allowed. The operators `within`, `contains` and `overlaps` are not specified in SQL99, but are introduced to specify the region on the sky as such search criterion is a fundamental one for the astronomical data service. These three operators compare two values of geometry data type, which is described in the next subsection.
4. Functions `point()`, `circle()` and `box()` are introduced to express the region on the sky, and `distance()` is also introduced to describe the proximity of the two coordinates.
5. Allowed logical operator is `AND` and `NOT`, and `OR` is not allowed as the mixture of `AND` and `OR` makes the search condition too complex.
6. Table and column may have an alias name.

2.2. Geometry Data Type

The geometry data type is introduced to represent a point or a region in the sky. A point is always expressed by two coordinates values, and in most of the cases a single coordinate value is meaningless and only the pair have physical meaning. So it is natural to have a column which has a pair of values to express the coordinate of celestial object, then it becomes simple to write the search criterion. For an example, if the coordinate values are prepared in two separate columns, the search condition is expressed as follows:

`point(t1.ra, t1.dec) within box((t2.ra, t2.dec), 1.0, 1.0)`. On the other hand, if the coordinate values are prepared in one single column, the same statement can be expressed shortly as follows: `t1.point within t2.region`. We defined “Point”, “Circle” and “Box” data types. Expressions of these data types in SQL are summarized in Table 1.

Table 1. Expressions of the geometry data type value

Data Type	Examples of Expression
Point	`Point(23., +10., 'FK5')`, `Point(23., +10.)`, `(23., +10.)`
Box	`Box((23.0, +10.0), 1, 1)`, `((23.0, +10.0), 1, 1)`
Circle	`Circle((23.0, +10.0), 1)`, `((23.0, +10.0), 1)`

2.3. Example SQLs for Basic Syntax

The next SQL shows an example of catalog data query.

```
Select g.ra, g.dec, g.mag_r
From   galaxy as g
Where  Point(g.ra, g.dec) within Circle((24.3, +5.0), 1.0)
       and g.mag_r < 24
```

This query describes “from a table named galaxy select right ascension, declination, and R-band magnitude of celestial objects located in the circle whose center coordinate is (ra, dec) = (24.3 deg, +5.0 deg) and radius is 1.0 deg”. In the “galaxy” table, coordinates of the objects are prepared in the two separate columns, “ra” and “dec”, so Point() function is required to describe the search region. In the “galaxy2” table of the next SQL sample, the coordinates are prepared in a single column “point”.

```
Select g.point, g.mag_r
From   galaxy2 as g
Where  g.point within Circle((24.3, +5.0), 1.0) and g.mag_r < 24
```

The next SQL shows an example of image data query.

```
Select img.filter, img.imageURL
From   imageData as img
Where  img.region = BOX((24.3, +5.0), 0.1, 0.1)
```

This equivalents to the next URL-based SIAP.

```
http://jvo.nao.ac.jp/image?POS=24.3,5.0&SIZE=0.1&FORMAT=VOTABLE
```

In the above SQL, the column `region` is used as a parameter to specify the region of interest, and the access URL of the cut out image is returned as a data of `imageURL` column. Those columns are actually not present in the relational table on the DBMS, but user can specify as if it virtually exists so these columns are called as a virtual column and its table is called as a virtual table or a view in the RDB terminology.

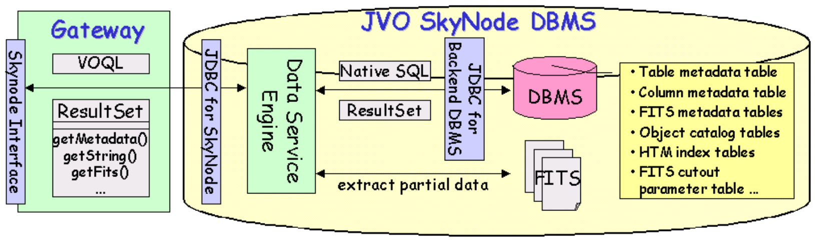


Figure 1. Architecture of JVO Skynode designed to be compatible with the proposed VOQL.

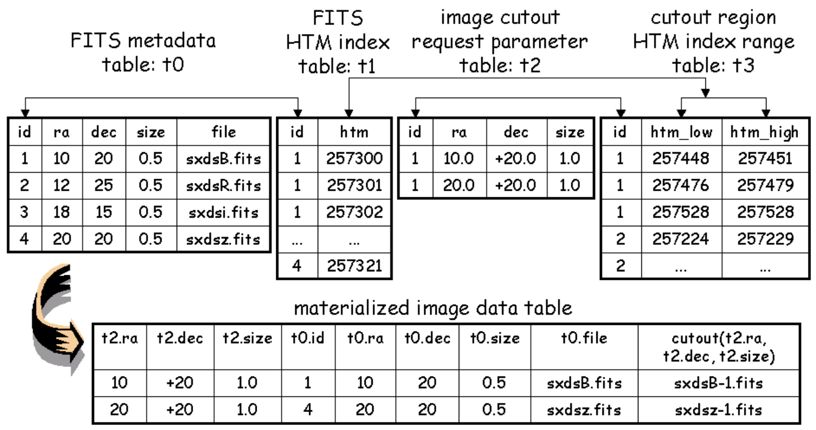
2.4. Syntax Enhancement

The following enhancement may be applied as an optional feature.

1. Multiple tables enhancement: Multiple tables can be queried with a single SQL. External tables provided in a form of VOTable must be treated in the same manner as the original tables of the data service. The external table is specified as `EXT::<fileNumber>.<resourceName>.<tableName>`. Join predicate and sub-query are not mandatory.
2. Unit support enhancement: Any numeric value may be followed by a unit and unit conversion must be carried out by the data service.
3. Algebraic expression enhancement: An algebraic expression can be specified in the selection list and search criteria in the "where" clause.
4. Logical operator `OR` enhancement: Support for mixture of `AND`, `NOT` and `OR` in the "where" clause.
5. Object data type enhancement: A column may have a structured data and access methods.
6. Use of identifier for specifying a table name (portal): A table name is expressed by an identifier of dot notation to specify the table uniquely in the VO. A dot character in the identifier must be escaped by a back slash. This is a feature dedicated to a portal data service.
7. UCD (portal): A UCD can be used in place of a column name. This is a feature dedicated to a portal data service. The portal service searches data resources which have UCD specified in the SQL and translate to the column name using the column meta data collected from the data services.
8. Omission of "From" part (portal): This is a feature dedicated to a portal data service. The portal searches the data resources from the registry according to the query condition. In this case, UCD must be used for describing the selecting column and the query condition.

3. JVO Skynode Architecture

We have designed an architecture to implement the data service compatible with the proposed query language (Figure 1). The "gateway" in the figure provides the interface to access the skynode (Yasuda et al. 2004) services. The received query is transfered to the JVO SkyNode DBMS and the result is returned in the form of a `ResultSet` java object. The access interface is provided by JDBC which is a standard interface to access the DBMS in the Java environment.


FITS metadata table: t0
FITS HTM index table: t1
image cutout request parameter table: t2
cutout region HTM index range table: t3
materialized image data table

id	ra	dec	size	file
1	10	20	0.5	sxdsB.fits
2	12	25	0.5	sxdsR.fits
3	18	15	0.5	sxdsi.fits
4	20	20	0.5	sxdsz.fits

id	htm
1	257300
1	257301
1	257302
...	...
4	257321

id	ra	dec	size
1	10.0	+20.0	1.0
1	20.0	+20.0	1.0

id	htm_low	htm_high
1	257448	257451
1	257476	257479
1	257528	257528
2	257224	257229
2	...	...

t2.ra	t2.dec	t2.size	t0.id	t0.ra	t0.dec	t0.size	t0.file	cutout(t2.ra, t2.dec, t2.size)
10	+20	1.0	1	10	20	0.5	sxdsB.fits	sxdsB-1.fits
20	+20	1.0	4	20	20	0.5	sxdsz.fits	sxdsz-1.fits

Figure 2. An example to materialize the virtual image table.

Celestial object catalog is stored in a relational data base (RDB) system, such as PostgreSQL, MySQL, Oracl, and so on, and FITS files are managed in the unix files system and its meta data is stored in the RDB. HTM index (Kunszt et al. 2000) is used to perform a quick data search. Query condition related to the search region on the sky is converted to a condition on the HTM index ranges at the "Data Service Engine", and then the modified SQL is submitted to the DBMS. In the case of image data query, it is necessary to materialize the virtual image table as follows (Figure 2): First, parameters specifying the image cut out regions are stored in a table (t2) and the corresponding HTM index range table (t3) is also created. Then FITS meta data table is joined with table t2 with intervention of the HTM index table of t1 and t3, and the virtual image table is materialized as shown in the bottom of the figure.

Acknowledgments. This work was supported by the JSPS Core-to-Core Program and Grant-in-aid Information Science (15017289 and 16016292) carried out by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

Yasuda, N. et al. 2004, in ASP Conf. Ser., Vol. 314, ADASS XII, ed. F. Ochsenbein, M. Allen, & D. Egret (San Francisco: ASP), p.293

P. Z. Kunszt, A. S. Szalay, I. Csabai, A. R. Thakar. 2000, in ASP Conf. Ser., Vol. 216, ADASS IX, ed. N. Manset, C. Veillet, & D. Crabtree (San Francisco: ASP), p.141