Restricted Geometry Support in ADQL

Walter Landry

TAP Geometry Backends at IRSA

- IRSA implements geometry for our TAP services with 2 different back ends

HTM

PostGIS
Geospatial Objects for PostgreSQL

TAP Geometry Backends at IRSA

- IRSA implements geometry for our TAP services with 2 different back ends

HTM

- Every other TAP service with geometry support uses pgSphere

HTM Implementation

- HTM gives a unique id for each triangle
- For our catalogs, which consist of (ra,dec) pairs, we add the HTM id and x,y,z coordinates.
- Then we rewrite queries to use those HTM id's and coordinates.

Rewriting Queries to use HTM

```
Contains(Point(ra,dec),
         Circle(ra_in,dec_in,r_in))=1
```

↓

```
((htm>htm_0 and htm<htm_1)
 or (htm>htm_2 and htm<htm_3)
 or (htm>htm_4 and htm<htm_5)
 or (htm_htm_6 and htm<htm_7))
and power(x-x_in,2) + power(y-y_in,2)
  + power(z-z_in,2)
  < 4*power(sin(r_in/2),2)
```

Rewriting Queries to use HTM

- This helps the database query optimizer run the query efficiently.
- It does mean that we need the shape at parse time.

Shape parameters specified inline (circle(13.5, -12.7, 0.01))	Y
Shape parameters from uploaded tables (mytable.ra, mytable.dec)	Y
Shapes in uploaded tables vary from row to row using REGION strings	N
Shape parameters come from a subquery	N

Enables non-SQL backends

- This approach does not require stored procedures that interact with SQL.

Enables non-SQL backends

- This approach does not require stored procedures that interact with SQL.
- With a specialized spatial database, you can run the geometric queries first.

Enables non-SQL backends

- This approach does not require stored procedures that interact with SQL.
- With a specialized spatial database, you can run the geometric queries first.
- Then pipe the results into a RDBMS (sqlite, mysql,...) and run the rest of the query.

Comparing this with ObsCore use cases

- Full geometry query is not specified in ObsCore standard.
- The following examples are from CADC.

`http://www.cadc-cdda.hia-iha.nrc.cnrc.gc.ca/cvo/ObsCore`

ObsCore A 1.1

```
SELECT * from ivoa.ObsCore
WHERE em_min < 2.48e-10 and em_max>2.48e-10
AND CONTAINS(POINT('ICRS',16,10),s_region)=1
AND t_exptime>10000
```

ObsCore A 1.1

```
SELECT * from ivoa.ObsCore
WHERE em_min < 2.48e-10 and em_max>2.48e-10
AND CONTAINS(POINT('ICRS',16,10),s_region)=1
AND t_exptime>10000
```

$$\downarrow$$

```
SELECT * from ivoa.ObsCore
WHERE CONTAINS(POINT('ICRS',16,10),s_region)=1
AND (em_min < 2.48e-10 and em_max>2.48e-10
     AND t_exptime>10000)
```

ObsCore A 1.1

```
SELECT * from ivoa.ObsCore
WHERE em_min < 2.48e-10 and em_max>2.48e-10
AND CONTAINS(POINT('ICRS',16,10),s_region)=1
AND t_exptime>10000
```

$\downarrow$

```
SELECT * from ivoa.ObsCore
WHERE CONTAINS(POINT('ICRS',16,10),s_region)=1
AND (em_min < 2.48e-10 and em_max>2.48e-10
     AND t_exptime>10000)
```

ObsCore A 1.1

```
SELECT * from ivoa.ObsCore
WHERE em_min < 2.48e-10 and em_max>2.48e-10
AND CONTAINS(POINT('ICRS',16,10),s_region)=1
AND t_exptime>10000
```

$\downarrow$

```
SELECT * from ivoa.ObsCore
WHERE CONTAINS(POINT('ICRS',16,10),s_region)=1
AND (em_min < 2.48e-10 and em_max>2.48e-10
     AND t_exptime>10000)
```

ObsCore A 1.2a

```
SELECT i.*, x.dataproduct_type, ...
FROM ivoa.ObsCore AS x
  JOIN TAP_UPLOAD.inputA as i
  ON CONTAINS(POINT('ICRS',i.ra,i.dec),
              x.s_region)=1
WHERE x.dataproduct_type='image'
  AND em_min < 1.0e-8 and em_max>5.0e-9
```

ObsCore A 1.2a

```
SELECT i.*, x.dataproduct_type, ...
FROM ivoa.ObsCore AS x
  JOIN TAP_UPLOAD.inputA as i
  ON CONTAINS(POINT('ICRS',i.ra,i.dec),
              x.s_region)=1
WHERE x.dataproduct_type='image'
  AND em_min < 1.0e-8 and em_max>5.0e-9
```

ObsCore A 3.5

```
SELECT * FROM ivoa.ObsCore
WHERE dataproduct_type='cube'
  AND (em_max-em_min)>0.599585
  AND 8.6696e-4 between em_min and em_max
  AND SQRT(AREA(s_region))/s_resolution>=100
```

ObsCore A 3.5

```
SELECT * FROM ivoa.ObsCore
WHERE dataproduct_type='cube'
  AND (em_max-em_min)>0.599585
  AND 8.6696e-4 between em_min and em_max
  AND SQRT(AREA(s_region))/s_resolution>=100
```

ObsCore A 3.9

```
SELECT * FROM ivoa.ObsCore
WHERE CONTAINS(POINT('ICRS',ra((t_max-t_min)/2),
                            dec((t_max-t_min)/2),
                            s_region)=1
```

Mostly compatible?

- This approach satisfies all of the listed use cases.
- There may be valid use cases which are not covered by this.

ADQL EBNF Modifications

```
<search_condition>::=
  {<predicate_geometry_function>
   [ AND <non_geometry> ]}
  | {<non_geometry>
     [ AND <predicate_geometry_function> ]}

<non_geometry>::=
  <left_paren>
    <old_search_condition>
  <right_paren>
```

Remove geometry from functions

```
<string_value_function>::=
  <user_defined_function>
  | <string_geometry_function>

<value_expression>::=
  <numeric_value_expression>
  | <string_value_expression>
  | <geometry_value_expression>
```

Simplify numeric geometric functions

```
<numeric_geometric_function>::=
  <non_predicate_geometry_function>

<area>::=AREA <left_paren>
  <column_region_reference> <right_paren>
```

- <column_region_reference> is a <column_reference> to a simple table (no subqueries) which must be a region with a fixed type for all rows.
- remove <distance>, <coord1>, <coord2>

Geometry Expression

```
<geometry_value_expression>::=
  <column_region_reference>
  | <geometry_value_function>

<coordinate>::= <upload_column_reference>
  | [<sign>] <unsigned_numeric_literal>
```

- <upload_column_reference> is a column in an uploaded table.
- <radius>, <box> sizes, and <region> strings are similarly constrained.

Geometry Predicates

- This means that geometric shapes can only be constructed from uploaded tables and literals.

Geometry Predicates

- This means that geometric shapes can only be constructed from uploaded tables and literals.
- Other tables already have appropriate regions (e.g. POINT's for catalogs, POLYGON's for images)

Geometry Predicates

- This means that geometric shapes can only be constructed from uploaded tables and literals.
- Other tables already have appropriate regions (e.g. POINT's for catalogs, POLYGON's for images)
- It strengthens the type system. You can not accidently write CIRCLE('ICRS',dec,ra,r) for built-in tables.

Geometry Predicates

- This means that geometric shapes can only be constructed from uploaded tables and literals.
- Other tables already have appropriate regions (e.g. POINT's for catalogs, POLYGON's for images)
- It strengthens the type system. You can not accidently write CIRCLE('ICRS',dec,ra,r) for built-in tables.
- Oddly enough, the current standard disallows

  CIRCLE(a.point,a.r)

  POLYGON(a.point0,a.point1,a.point2)