Versatile access to HEALPix-based sky region objects within PostgreSQL data bases with PgSphere

Markus Nullmeier

Zentrum für Astronomie der Universität Heidelberg
Astronomisches Rechen-Institut

mnullmei@ari.uni.heidelberg.de

Versatile access to HEALPix-based sky region objects within PostgreSQL data bases with PgSphere

Markus Nullmeier

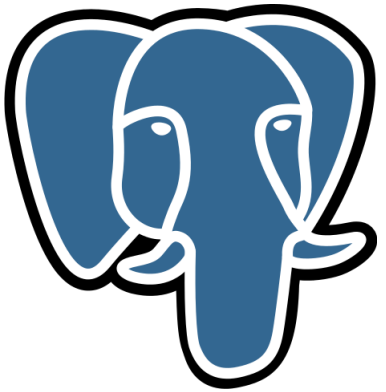
Zentrum für Astronomie der Universität Heidelberg
Astronomisches Rechen-Institut

mnullmei@ari.uni.heidelberg.de

- **Overview: retrieving spherical objects with pgSphere**
- **Requirements for sky region objects and their implementation**

About PgSphere

- **PostgreSQL**
  “The world's most advanced open source database”

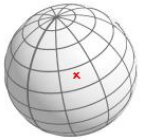


- **Store and retrieve data living on the celestial sphere**

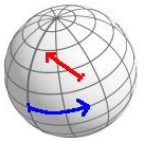
- **PostgreSQL extension:**
  SQL data types, functions, indexes

PgSphere: spherical data types

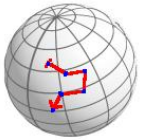
- Spherical points (RA, DEC)



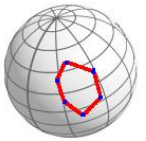
- Spherical lines



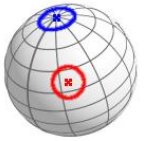
- Spherical paths



- Spherical polygons

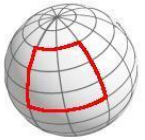


- Spherical circles

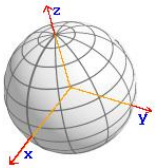


- Spherical ellipses

- Spherical coordinate ranges



- Spherical transformations (rotations)



Pgsphere demonstration

```
psql=# create extension pg_sphere;

psql=# create table my_table (name varchar, ra_dec spoint);
psql=# insert into my_table values ('Aldebaran',  '(4h 35m 55.24s, +16d 30m 33.49s)');
psql=# insert into my_table values ('Betelgeuse', '(5h 55m 10.31s,  +7d 24m 25.43s)');
psql=# insert into my_table values ('Antares',   '(16h 29m 24.46s, -26d 25m 55.21s)');

psql=# select length(spath(ra_dec)) from my_table;
      length
-----------------
 3.02828754

psql=# select * from my_table where ra_dec <@  scircle '<(80d, 10d), 15d>';

    name    |           ra_dec
------------+----------------------------
 Aldebaran  | (68.98017d , 16.5093d)
 Betelgeuse | (88.79296d , 7.407064d)
```

Pgsphere demonstration

```
psql=# create extension pg_sphere;

psql=# create table my_table (name varchar, ra_dec spoint);
psql=# insert into my_table values ('Aldebaran',  '(4h 35m 55.24s, +16d 30m 33.49s)');
psql=# insert into my_table values ('Betelgeuse', '(5h 55m 10.31s,  +7d 24m 25.43s)');
psql=# insert into my_table values ('Antares',   '(16h 29m 24.46s, -26d 25m 55.21s)');

psql=# select length(spath(ra_dec)) from my_table;
      length
-----------------
 3.02828754
```

utility functions

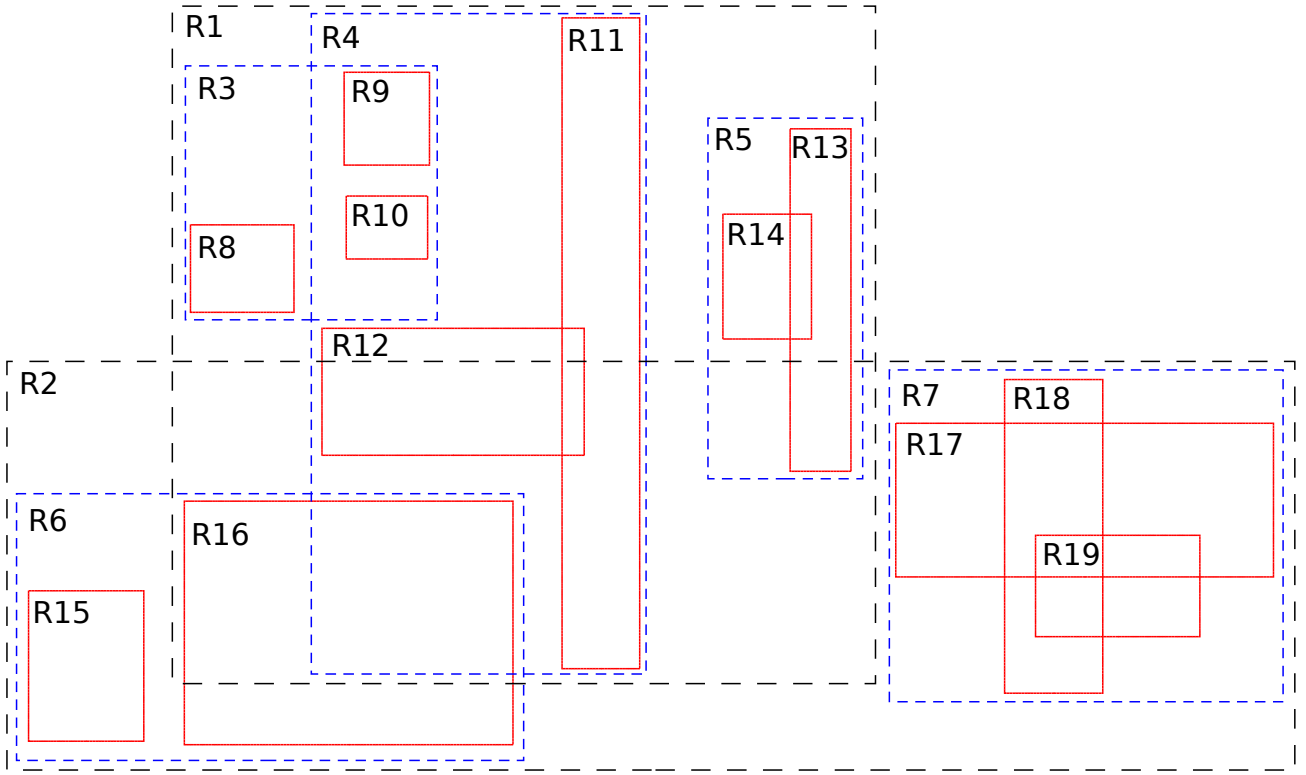
```
psql=# select * from my_table where ra_dec <@  scircle '<(80d, 10d), 15d>';

    name    |           ra_dec
------------+----------------------------
 Aldebaran  | (68.98017d , 16.5093d)
 Betelgeuse | (88.79296d , 7.407064d)
```

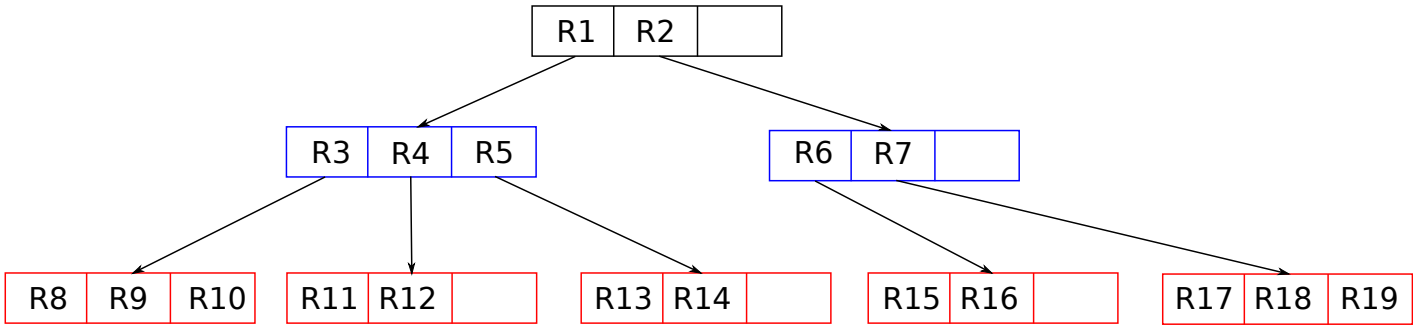
cone search

— accerleated by indexes

Pgsphere internals

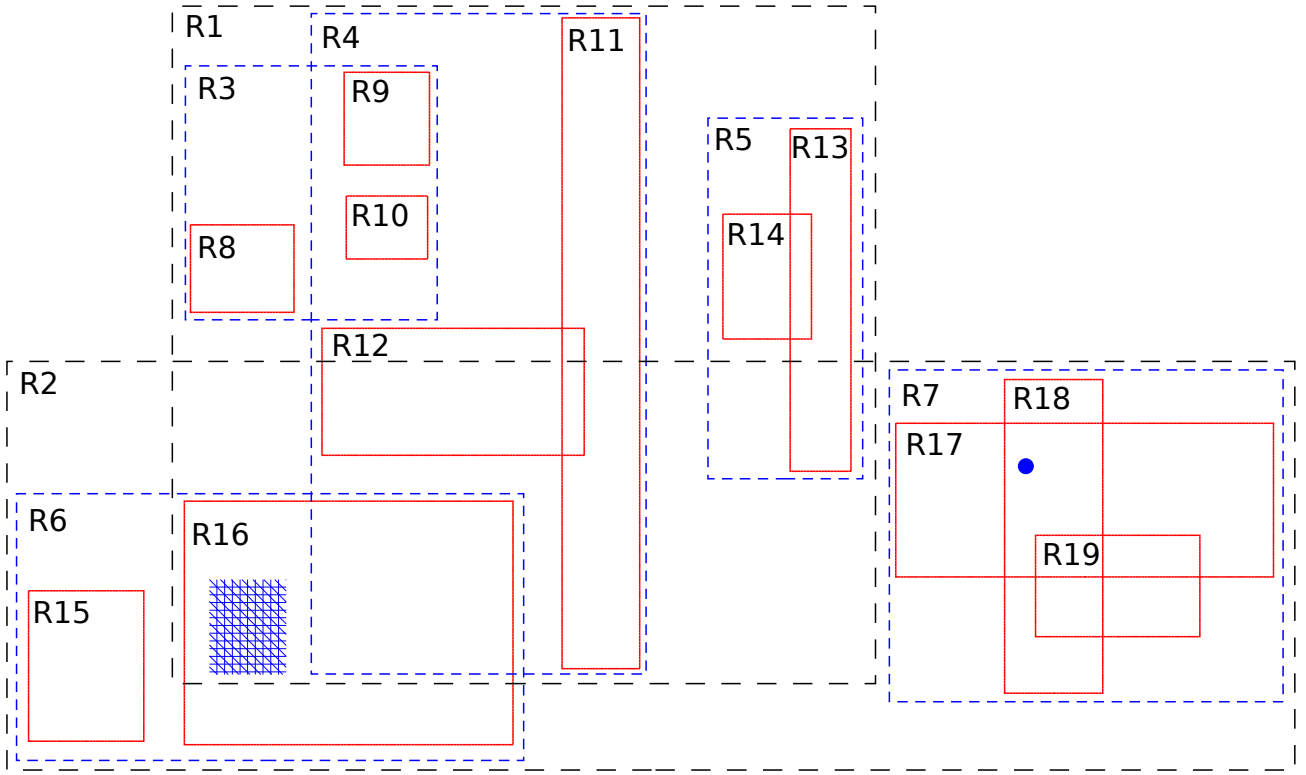
R-tree

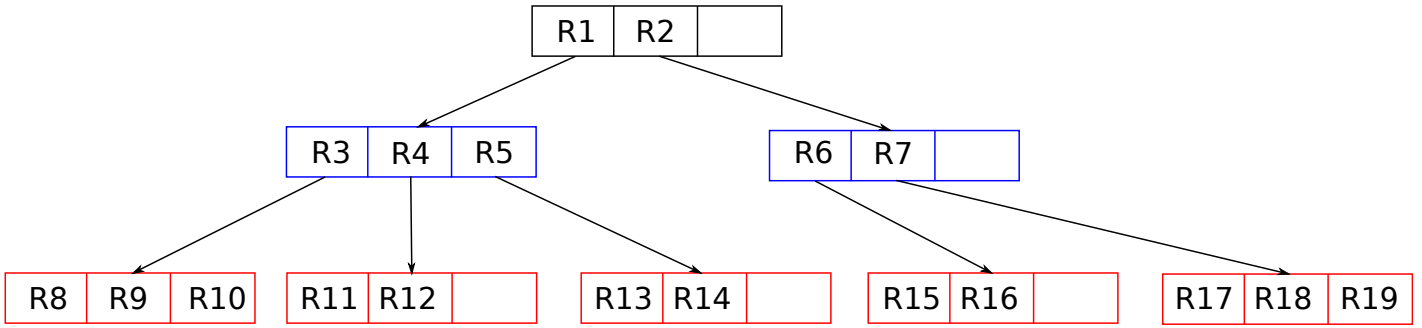




Pgsphere internals

R-tree





Pgsphere development history

Janko Richter

Teodor Sigaev Oleg Bartunov

Igor Chilingarian

Pgsphere development today

Dmitry Ivanov

Alexander Korotkov

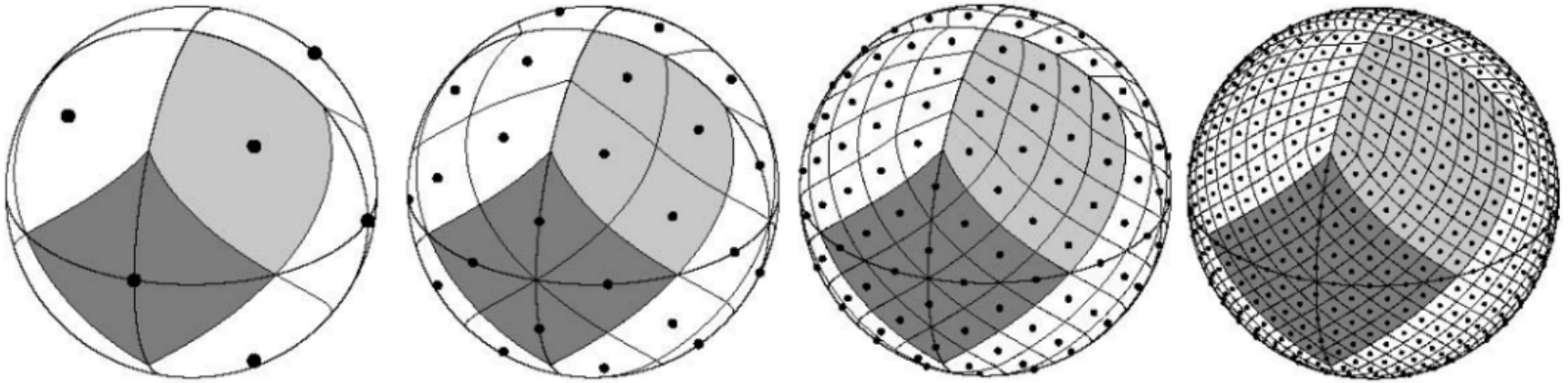
Markus Nullmeier

contributors: Pat Dowler, Serge Monkewitz

http://pgsphere.github.io

Requirements for sky region objects (I)

- **Based on HEALPix**

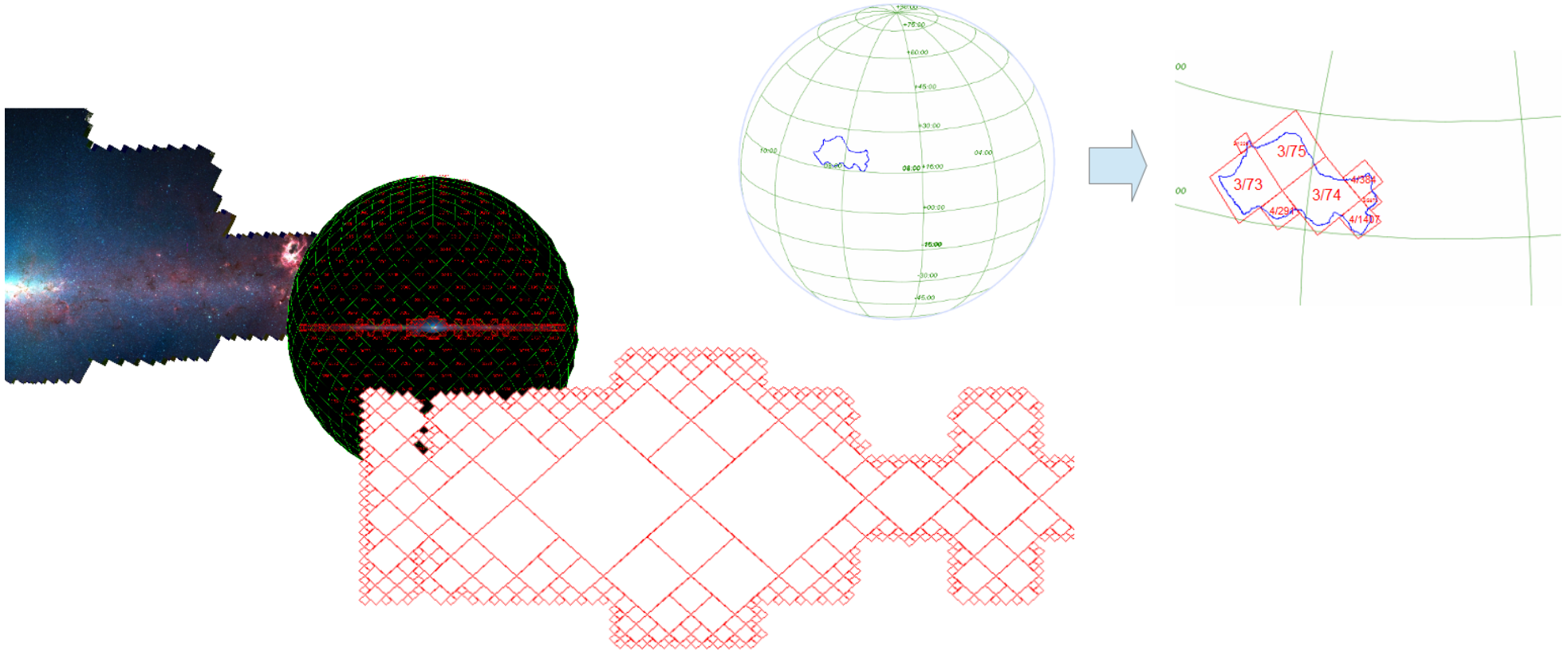


- **Start with 12 diamonds**
- **Subdivide by fours**

Requirements for sky region objects (II)

MOC = Multi-order coverage (HEALPix Multi-Order Coverage map)

- **Concise mapping of a catalog's coverage of the sphere**

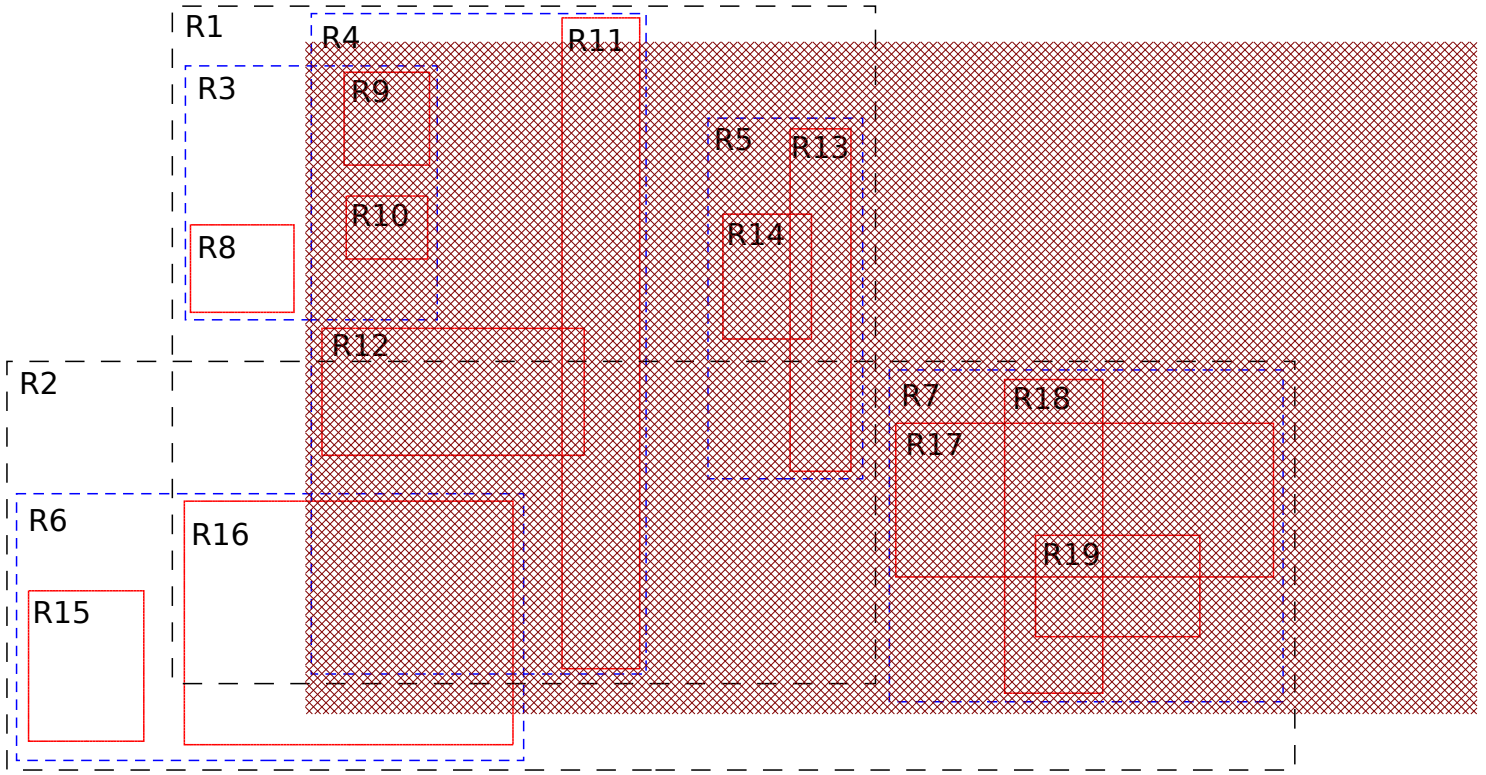


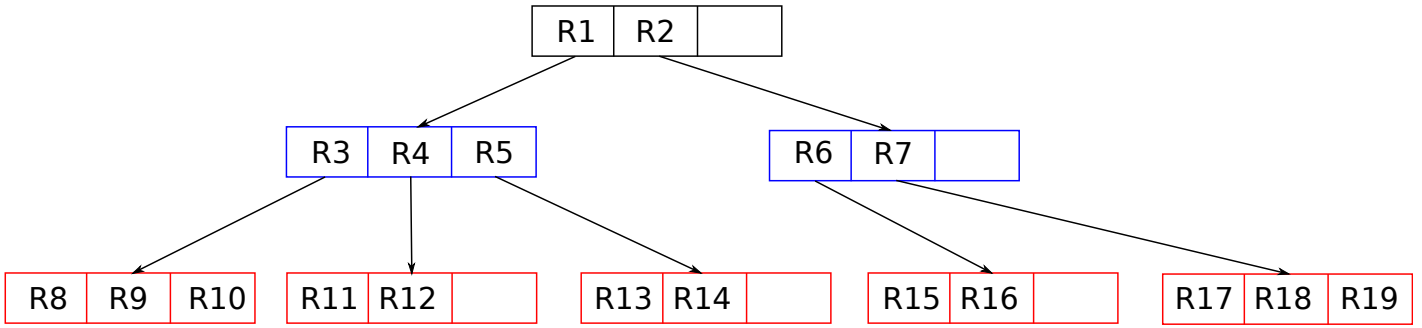
- **MOC is an IVOA standard**

Requirements for sky region objects (III)

- **Utility functions / operators for, e. g., contains queries**
- **Fast access to objects spanning multiple PostgreSQL pages:**
  – *internally store a sky region object as a (read-only) B-tree*
- **Efficient representation of multiple HEALPix levels as *intervals on the finest level***
- **Create sky region object from table columns or queries**
- **Indexes for columns of sky region objects**

Indexing: R-trees fail for for large objects





Sky region indexing (I)

- Comes down to well-known set indexing problem
- Set elements are intervals of HEALPix numbers
- Most basic set indexing technique: “inverted files”
- Use modified B-tree index, customised from a loadable indexing module for PostgreSQL 9.6+ (https://github.com/postgrespro/rum)

Sky region indexing (II)

- **Internal logic of indexing with an “inverted file” stucture:**

sorted intervals of HEALPix numbers	Sets of “pointers” to sky region objects
interval0	{ obj7, obj11 }
interval1	{ obj2, **obj108**, obj109 }
interval2	{ **obj108**, obj732, obj11030 }
...	...

- **Accelerated operations:**

containment, union, cross section, ...

Use cases for sky region objects

- **Create the sky coverage of a given database table “my_cat”:**

  SELECT smoc(my_cat.ra_dec, level = 15) FROM my_cat ;

- **Search all catalogs that fully contain the sky region of a user-specified MOC region:**

  SELECT name FROM catalogs
  WHERE my_moc <@ catalogs.moc ;

Interested? Your involvement:

- Send in feature requests
- Soon, download the code from `https://github.com/mnullmei/ivoa_moc`
- Send in bug reports
- Send in test cases
- Send in patches