Observatoire astronomique de Strasbourg

H3C : librairie HEALPix pour PostgreSQL

H3C - Généralités

H3C : **H**EALPix **T**ree **C**

- Index fonctionnel btree PostgreSQL :

  CREATE index indexname **ON** table(healpix(ra,dec)) ;

  SELECT * **FROM** table **WHERE** healpix(ra,dec)=...

- H3C est basée sur Q3C (Koposov)
- Contient une librairie de fonctions qui utilisent l'index HEALPix.

  Conesearch, polygon, jointures....
- Utilisé dans TAPVizieR qui contient une couche Java ADQL pour transformer ADQL → SQL

(extension P3CTranslator de la classe PostgreSQLTranslator de la libairie tapLib de Gregory)

H3C - Généralités

Quelques fonctions utiles :

h3c_ang2ipix(ra, de [, nside]) RETURN the HELPix number from a position
h3c_ipix2ang(ipix [,nside]) RETURN the center of the HEALPix cell

h3c_radial_query(ra1, de1, ra0, de0, radius [,nside])
make a cone search arround position (ra0, de0)
h3c_join(ra1, de1, ra2, de2, radius [,nside])
join 2 tables according to their positions

h3c_in_poly(ra, de, Array()) RETURN True if (ra,de) is in the polygon
h3c_poly_query(ra, de, Array() [, nside]) RETURN True if (ra,de) is in a polygon
(available for convex only)

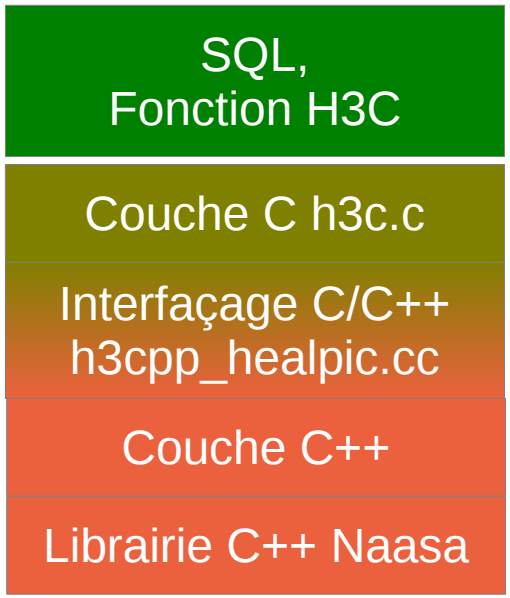
h3c_in_ellipse(ra, de, ra0, de0, may_ax, axis_ratio, position_angle)
RETURN if (ra, de) is in the ellipse
the function doesn't use index

h3c_dist(ra1, de1, ra2, de2) RETURN the distance between 2 points

Observatoire astronomique de Strasbourg

H3C- Architecture

- H3C utilise la librairie 64bits C++ de la nasa (Martin Reineke)
  - Librairie C trop pauvre, librairie C++ moins riche que Java
  - A nécéssité des modifications pour optimiser les recherches sur de petits cônes dans le cas de crossmatch



H3C - développement

Exemple : la procédure stockée h3c_ang2ipix :

```
CREATE OR REPLACE FUNCTION h3c_ang2ipix(double precision, double
                                        precision, integer)
        RETURNS bigint
        AS 'MODULE_PATHNAME', 'pgh3c_ang2ipix'
        LANGUAGE C IMMUTABLE STRICT;
```

Observatoire astronomique de Strasbourg

H3C - développement

```
00124 PG_FUNCTION_INFO_V1(pgh3c_ang2ipix);
00125 Datum pgh3c_ang2ipix(PG_FUNCTION_ARGS)
00126 {
00127         h3c_coord_t ra = PG_GETARG_FLOAT8(0);
00128         h3c_coord_t dec = PG_GETARG_FLOAT8(1);
00129         int nside = PG_GETARG_UINT32(2);
00130         h3c_ipix_t ipix;
00131         static int invocation;
00132         static h3c_coord_t ra_buf, dec_buf;
00133         static h3c_ipix_t ipix_buf;
00134         static int nside_buf;
00135
00136        if ((ra == ra_buf) && (dec == dec_buf) && nside == nside_buf) {
00144                     PG_RETURN_INT64(ipix_buf);
00146         }
00147
00148         h3c_ang2pix_nest(nside, h3c_dec2theta(dec*H3C_DEGRA),
00150                                 h3c_ra2phi(ra*H3C_DEGRA),  (h3c_ipix_t *) &ipix);
00152
00153         ra_buf = ra;
00154         dec_buf = dec;
00155         ipix_buf = ipix;
00156         nside_buf=nside;
00158         PG_RETURN_INT64(ipix);
00159 }
```

Cas du du cone search:

h3c_radial_query recherche toutes les cellules HEALPix qui recouvre le cône.

Une procédure ne prends pas en compte l'index!
→ La liste n'est pas utilisée directement

→ Utilisation de couples [borne inf, borne max] pour décomposer la requête:

- l'idée étant d'utiliser la fonction h3c_ang2ipix pour forcer l'index
- d'utiliser un itérateur (ou la liste HEALPix est calculée au 1er passage)

H3C - développement

```
CREATE OR REPLACE FUNCTION h3c_join( double precision, double precision,
      double precision, double precision, double precision)     returns boolean as 'SELECT((
(h3c_ang2ipix($3,$4)>=h3c_nearby_it($1,$2,$5,0) AND
h3c_ang2ipix($3,$4)<=h3c_nearby_it($1,$2,$5,1)) OR
(h3c_ang2ipix($3,$4)>=h3c_nearby_it($1,$2,$5,2) AND
h3c_ang2ipix($3,$4)<=h3c_nearby_it($1,$2,$5,3)) OR
(h3c_ang2ipix($3,$4)>=h3c_nearby_it($1,$2,$5,4) AND
h3c_ang2ipix($3,$4)<=h3c_nearby_it($1,$2,$5,5)) OR
(h3c_ang2ipix($3,$4)>=h3c_nearby_it($1,$2,$5,6) AND
h3c_ang2ipix($3,$4)<=h3c_nearby_it($1,$2,$5,7))
)
 AND h3c_sindist($1,$2,$3,$4)<POW(SIN(RADIANS($5)/2),2)
)
' LANGUAGE SQL IMMUTABLE;
```

H3C-développement

1) Décision de l'odre (nside) en fonction du rayon

radius < 2min	ordre=15, nside=32768 (entier long)
2min < radius < 0,5 deg	ordre=13
1 deg < radius < 15 deg	ordre=11
...	

2) Création de la liste des cellules qui recouvrent le cône

(void Healpix_Base::query_disc_inclusive)

3) Tri et regroupement des cellules dans des intervalles.

liste healpix → [(min0,max0), (min1,max1), ..., (minp,maxp)]

p=100 pour un conesearch, =4 pour un crossmatch)

4) Les bornes (mini,maxi) sont remis dans l'ordre (nside) initial

Cas du pôlygone :

- pas de fonctions polygones dans la librairie c++
- UNIQUEMENT CONVEXE!

Idée : décomposition en sous requêtes (i.e. : conesearch) avec utilisation d'un itérateur et de la fonction indexée h3c_ang2ipix

1) une fonction qui permet de savoir si un point est dans un polygone (produit vectoriel)
2) recherche des HEALPix dont le centre est dans le polygone + les cellules dont la distance du centre et des vertex est < rayon maximum du cercle circonscrit pour un ordre donné.
   - décision de l'ordre (nside)
   - balayage récursif limité au cercle circonscrit du polygône
   - tri et regroupement des cellules HEALPix, …..

H3C - configuration

Contrainte PostgreSQL et tuning (décrit dans la doc)

1) l'ordre des tables dans la fonction h3c_join est important
2) Forcer l'utilisation de l'index
   - dans la session: set enable_seqscan=false
     set enable_nestloop=false

Postgresql.conf

- forcer l'utilisation de l'index pour la base de donnée "seq_page_cost" et "random_page_cost"
- augmenter la mémoire cache
  - augmenter la mémoire cache utilisé par le plan de requete: effective_cache_size = 2048MB
  - augmenter la mémoire cache du dique : shared_buffers = 2048Mb

H3C-développement

Librairie téléchargeable, libre et documentée.

http://cds.u-strasbg.fr/resources/doku.php?id=h3c