MCS & Co.: a customizable server and database management system for Astronomy

L. Nicastro, INAF – IASF Bologna, I

- Principles and implementation
- MCS classes examples
- VOTPP: VOTable Parser library
- MyRO: row-based tables access
- DIF: indexing the sky
- A REM-like (multi-instrument) images archive manager

- DIF indexed catalogues in Turin

Beijing, 11-07-2011

The MCS (My Customizable Server) paradigm

- ✧ Client-server model:
  1. multi-thread, 2. information server, 3. database server
  - – It takes care of all low-level interactions with networking, threading, and database code
- ✧ Data abstraction layer
  - – Automatic management of data types conversion
- ✧ Multi-language client interface
  - – Functions call standardization ⇒ clients use more convenient language
- ✧ External tools integration
  - – Can reuse or create external stand-alone tools and retrieve output
- ✧ Messages, data objects (records) and file transfer facility via custom protocol
- ✧ Client/server application parameters setting via INI file
- ✧ SSL/TSL (secure socket layer) support

Additional tools: **MyRO** – Privilege system at the record level for MySQL tables,
DIF – Automatic indexing of MySQL tables with spherical coordinates,
VOTPP – VOTable and FITS file handling

Beijing, 11-07-2011

MCS technically speaking

A collection of high level C++ classes aimed at implementing:

- Multi-thread applications
- Network applications (via TCP)
- Database applications (MySQL)
- Information servers

additionally it also implements:

- Client interfaces for (*almost*) all languages (e.g. IDL, Fortran)
- VOTable and FITS file access ⇒ ***VOTPP***
- Privilege system at the record level for MySQL tables ⇒ ***MyRO***
- Automatic indexing of MySQL tables with spherical coordinates ⇒ ***DIF***

MCS and MySQL strongly linked (at the moment), why?

- *Free, open-source, tested, familiar (to me)*
- *Fast, reliable, flexible:* triggers, views
- *plug-ins* ⇒ DB engines, *UDFs* ⇒ Expandible

See: ross.iasfbo.inaf.it/MCS

Beijing, 11-07-2011

MCS vs shell comparison

- ✧ The MCS account == MySQL account.
- ✧ Users can connect to a MCS server from any TCP/IP capable application, for example “telnet”.
- ✧ The MCS multi-language client library enables applications written in any language to interact with:
  1. the server,
  2. the (MySQL) DB server.

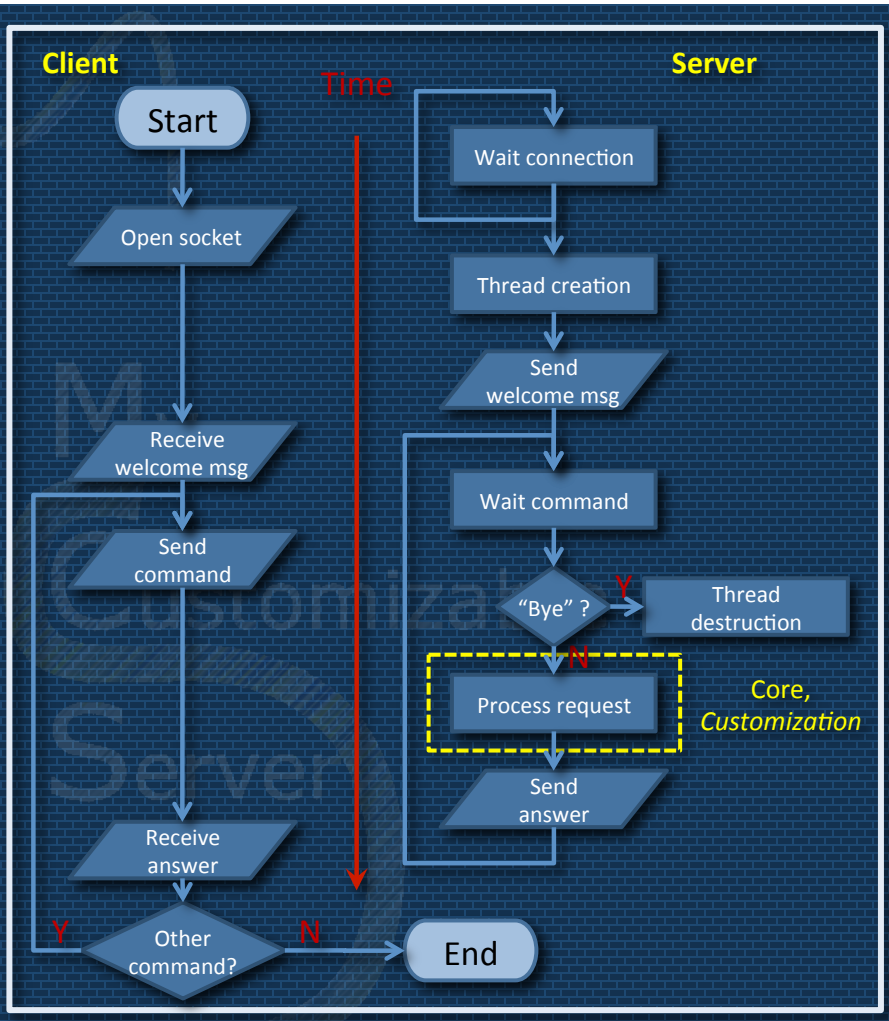
  Note: *in the latter case there is no need to have a MCS sever running.*

Unix shell		MCS server
stdin & stdout	⇒	bidirectional TCP socket
system account	⇒	MySQL account
internal commands	⇒	base commands
programs, shell scripts	⇒	external programs (EXEC command)
home directory	⇒	work directory

Beijing, 11-07-2011

MCS client-server flow diagram

⇒ *Only the server has to be written in C++! Client applications can be written in any language.*



Simplest MCS server

```
#include <mcs.hh>
using namespace mcs;

int main() {

  mcsStart("simplest");  // Start the server daemon

}
```

⇒ *Default port* 6523

MCS client: C++

```
#include <mcs.hh>
using namespace mcs;

int main() {
// Connect to the MCS server
  Client cli("./", "localhost", 6523);
// Perform authentication
  cli.login("mcsuser", "mcspass", "test");
// Upload a file
  cli.exec("PUT myfile");
// Download a file
  cli.exec("GET myfile");
// Execute a query on remote database
  cli.exec("QRY SELECT * FROM mcstest");
  cli.exec("QRES");
```

```
// Loop through the resulting record set
  while (! cli.eof()) {
   // Get a reference to current record
   Record& rec = cli.rec();

   // For each field print its value
   for (int i=0; i<rec.count(); i++)
    cout << rec[i].sval() << "\t";
   cout << endl;

   // Move to next record
   cli.setNext();
  }
}
```


MCS client: C++

```
#include <mcs.hh>
using namespace mcs;

int main(int argc, char *argv[]) {
  //Connect to the MCS server
  Client cli("./", "localhost", 6523);

   //Perform authentication
  cli.login("mcstest", "mcstest", "test");

  //Upload a file
  cli.exec("PUT myfile");

   //Download a file
  cli.exec("GET myfile");

  //Execute a query on remote database
  cli.exec("QRY SELECT * FROM mcstest");
  cli.exec("QRES");

  //Loop through the resulting record set
  while (! cli.eof()) {

    //Get a reference to current record
   Record& rec = cli.rec();
```

```
   //For each field print its value
   for (int i=0; i<rec.count(); i++)
    cout << rec[i].sval() << "\t";
   cout << endl;

   //Move to next record
   cli.setNext();
  }
}
```

MCS client: Fortran

```
…
cli = new_Client(null, "", "localhost", 9001, 0)
dummy = Client_login(cli, "UserName", "Password", "DBname")

dummy = Client_exec(cli, "qry SELECT
                          camera, fname, date_obs, exposure
                          FROM ImgsLog", null)
nrows = RecordSet_nRows(cli)
…
cmd = "histo " // rfilename // ".fits.gz"
dummy = Client_exec(cli, cmd, null)

cmd = "GET " // rfilename // "_h.gif"
dummy = Client_exec(cli, cmd, null)
…
```


DB related classes: DBConn, Query (Table)

```
#include <mcs.hh>
using namespace mcs;

int main() {
//Connect to the database server
  DBConn db;
  db.connect("username", "password", "dbname");

//Execute a query through the opened connection
  Query qry(&db);
  qry.prepare("SELECT * FROM mcstest");
  qry.execute();

//Loop through the resulting record set
  while (! qry.eof()) {
//Get a reference to current record
    Record& rec = qry.rec();
//For each field print its value
    for (int i=0; i<rec.count(); i++)
      cout << rec[i].sval() << "\t";
    cout << endl;

//Move to next record
    qry.setNext();
  }
}
```

DB-MCS: Fortran interface + usrlib

```
PROGRAM MCSDEMO
IMPLICIT NONE

INCLUDE 'mcs_usrlib.inc'  ! ← user contributed lib
INTEGER*4 status

DATA c_user /'generic'/, c_password /'password'/,
.    c_dbname /'MyCats'/

c_query = 'SELECT htmID,GSChtm FROM regions'

status = DBExec_qry( 'localhost' )

CALL FieldInfo( 1 )

CALL printTable()

CALL DBFree()
END
```

```
INTEGER*8 vec, dbc, dbq, null  ! ← 32/64-bit
CHARACTER*16 idname

null = ifd_null()
dbc = new_DBConn( null )
CALL DBConn_connect(dbc, c_user, c_password,
.    c_dbname, c_host)
dbq = new_Query( null, dbc, 0 )
CALL Query_prepare( dbq, c_query )
CALL Query_execute( dbq, 1 )
IF (ifd_got_error() .EQ. 1) THEN
  WRITE(*,*) ifd_last_error()
  STOP
END IF
n_rows = RecordSet_nRows( dbq )
n_fields = RecordSet_nFields( dbq )

DO 10 i=1,n_rows
  vec = RecordSet_rec( dbq )
  id = Data_ival( Record_field(vec, 0) )
  idname = Data_sval( Record_field(vec, 1) )
  WRITE(*, '(I10, 2X, A)') id, idname
  dummy = RecordSet_setNext( dbq )
10  CONTINUE
```


DB-MCS: IDL interface

```
@mcs_usrlib  ; ← user contributed Functions and Pro

PRO mcsdemo, Host=host, User=user, Password=pass,
DBName=db, Query=query

COMMON Dbinfo
COMMON TABinfo

status = DBExec_Qry( db, user, pass, query, Host=host)
IF (status EQ 0) THEN BEGIN
  PRINT, "Fields info from FieldInfo:"
  FieldInfo, /INFO
  nf = TabFieldNames( 'regions', fn )
  PRINT, "Fields info from TabFieldNames:", fn
  nf = TabFieldTypes( 'regions', ft )
  PRINT, "Fields info from TabFieldTypes:", ft
  FieldNames, /INFO

; Build and fill the table structure 'tab_stru' as a
; structure of arrays
  TabStruct, /ARR_STRUCT, Debug=debug
  TabFill
ENDIF

DBFree
END
```

```
COMMON DBptr, dbc, dbq
COMMON DBinfo, c_host, c_user, c_password,
                c_dbname, c_query
COMMON TABinfo, mcs_types, is_rowstru, n_fields,
        n_rows, f_name, f_atype, f_itype, tab_stru

dbc = new_DBConn( ifd_null() )
DBConn_connect, dbc, c_user, c_password,
 c_dbname, c_host
dbq = new_Query( ifd_null(), dbc, 0 )
Query_query, dbq, c_query, 1

IF (Query_gotRecordSet( dbq )) THEN $
  nrec = RecordSet_nRows( dbq ) $
ELSE $
  nrec = Query_nAffectedRows( dbq )
n_fields = RecordSet_nFields( dbq )
v = RecordSet_metarec( dbq )
FOR j=0, n_fields-1 DO BEGIN
  d = Record_field( v, j )
  f_name[j]  = Data_name( d )
  t = Data_type( d )
  is_uns = Data_isUnsigned( d ) ...
Query_close, dbq
del_Query, dbq
del_DBConn, dbc
```

DB-MCS: IDL interface – HEALPix maps

```
@mcs_healplib   ; ← Include MCS-Base and MCS-HEALPix Functions/Programs

PRO mcshpxdemo, Host=host, User=user, Password=pass, DBName=db, Query=query

; Used to pass the map parameters
  COMMON HEALPmap, hp_nested, hp_nside, hp_order, hp_npix, $
                   hp_emptypix, hp_npixok, hp_map_nocc, hp_map_sum

  hp_order = 8
  query = 'select healpID_0_8 from UCAC_2orig'
  status = DBExec_Qry( db, user, pass, query )
  IF (status EQ 0) THEN BEGIN
    healp_MapFill()
    orthview, hp_map_nocc, rot=[45,45], coord=['C','G'], /grat, col=5
  ENDIF

; Eventually save the map into a FITS file
  Healp_MapSave( Outfile='UCAC2_objdens.fits' )

  DBFree
END
```

Beijing, 11-07-2011

MCS client: Python interface (1)

```
from python2mcs import *

null = ifd_null()
cli = new_Client(null, "./", "localhost", 9001, 0)
Client_login(cli, "username", "password", "dbname")

print "mcs server information returned by clinfo: "
Client_exec(cli, "clinfo", null)
printTable(cli)

print "\n"
Client_exec(cli, "qry SELECT camera, filename, dateobs, exposure, LEFT(filename, INSTR(filename, '.')-1) as froot
                  FROM Mainsrv_log", null)
nrows = RecordSet_nRows(cli)
print "Number of returned rows from Mainsrv: ", nrows
printTable(cli)

RecordSet_setFirst(cli)
for i in range(nrows) :
   print "\n"
   RecordSet_setPos(cli, i)
   p = RecordSet_rec(cli)
   Record_setFieldMap(p, "froot")
   f = Data_sval( Record_field(p, 0) )
```

(...continue)

Beijing, 11-07-2011

MCS client: Python interface (2)

```
    print "Retrieving sexcat file for ", f
    cmd = "sexcat " + f + ".fits.gz"
    Client_exec(cli, cmd, null)
    cmd = "get " + f + ".fits.gz.sexcat"
    Client_exec(cli, cmd, null)

    cmd = "medw " + f + ".fits.gz.sexcat"
    Client_exec(cli, cmd, null)
    ff1 = Data_fval( Record_field( Client_recv(cli), 0 ) )
    ff2 = Data_fval( Record_field( Client_recv(cli), 1 ) )
    print "Median Width and Height (pix.): ", ff1, ff2
if ifd_got_error() :
    print ifd_last_error()
ifd_reset_error()
del_Client(cli)
```

```
def printTable(cli):
  i = -1
  while RecordSet_eof(cli) == 0:
    i = i + 1
    p = RecordSet_rec(cli)

    if i == 0 :
      for jj in range(Record_count(p)) :
        d = Record_field(p, jj)
        print Data_name(d), "\t",
        continue
      print ""

    for jj in range(Record_count(p)) :
      d = Record_field(p, jj)
      print Data_sval(d), "\t",
    print ""

    RecordSet_setNext(cli)
    continue

  return 1
```


MCS Client, DBConn, Query, … what else?

- URLReader class ⇒ Pipe *(comm. between two threads) derived class*
- FITSReader class ⇒ RecordSet *derived class*

Example usage:

FITSReader fits
fits.open(ffile)
fits.HDUCount()
fits. selectHDU(i) or
fits. selectHDU("EXTANAME")
fits.header.count()
fits.header[i].name()
fits.header[i].sval() or
fits.header["BITPIX"].sval()
.ival() .lval() .fval() .dval() .tval()

fits.header_comments[i].sval()
…

fits.metarec().count()
fits.metarec().field(i).name()
fits.metarec().field(0).type()

fits.metarec().asStringNames()
fits.metarec().asStringTypes()
fits.rec().asString()
…

⇒ To import a FITS table into a MySQL table its enough to run:

```
fits2dbt File_name
```

VOTPP: C++ Parser Library

- A collection of high level (C++) classes and functions aimed to parse VOTables.
  - A VOTable file can be read using the `votpp::Parser_Stream` or the `votpp::Parser_Tree` class. In the former case it uses the SAX model (read one node at a time), in the latter the DOM model is used (read the entire file and build a browsable tree in memory).
  - VOTPP provide a distinct class for each possible VOTable node, so that the C++ code will be very close to the XML counterpart.
  - VOTable files can be read using the MCS's data abstraction layer.
  - FITS files base64 encoded or URL link ➔ read via CURL lib.

VOTPP: C++ Parser Library

VOTable classes ⇒ For each table node there is a class

- VOT_Element
- VOT_Binary
- VOT_Column
- VOT_Coosys
- VOT_Data
- VOT_Definitions
- VOT_Description
- VOT_Field
- VOT_FieldRef
- VOT_Fits
- VOT_Group
- VOT_Info
- VOT_Link
- VOT_Max
- VOT_Min
- VOT_Option
- VOT_Param
- VOT_ParamRef
- VOT_Resource
- VOT_Row
- VOT_Stream
- VOT_Table
- VOT_Tabledata
- VOT_Values
- VOT_VOTable
- VOT_Parser_Stream
- VOT_Parser_Tree
- *(VOT_Writer_Stream)*

VOTPP: C++ Parser Library

VOT_VOTable →
VOT_Resource →
VOT_Param →
VOT_Field →
VOT_Data →
VOT_Row →

```
<?xml version="1.0"?>
<VOTABLE version="1.1">
 <COOSYS ID="J2000" equinox="J2000." epoch="J2000." system="eq_FK5"/>
 <RESOURCE name="myFavouriteGalaxies">
  <TABLE name="results">
   <DESCRIPTION>Velocity and Distance estimations</DESCRIPTION>
   <PARAM name="Telescope" datatype="float" ucd="phys.size;instr.tel"
          unit="m" value="3.6"/>
   <FIELD name="RA"   ID="col1" ucd="pos.eq.ra;meta.main" ref="J2000"
          datatype="float" width="6" precision="2" unit="deg"/>
   <FIELD name="Dec"  ID="col2" "pos.eq.dec;meta.main" ref="J2000"
          datatype="float" width="6" precision="2" unit="deg"/>
   <FIELD name="Name" ID="col3" ucd="meta.id;meta.main"
          datatype="char" arraysize="8*"/>
   <DATA>
    <TABLEDATA>
     <TR>
      <TD>010.68</TD><TD>+41.27</TD><TD>N  224</TD>
     </TR>
     <TR>
      <TD>287.43</TD><TD>-63.85</TD><TD>N 6744</TD>
     </TR>
    </TABLEDATA>
   </DATA>
  </TABLE>
 </RESOURCE>
</VOTABLE>
```

VOT_Column

VOTPP: C++ Parser Library

Access to a node of type FIELD:

```
<FIELD
  name = "RA"
  ID = "col1"
  ucd = "pos.eq.ra;meta.main"
  ref = "J2000"
  datatype = "float"
  width = "6"
  precision = "2"
  unit = "deg"
/>
```

```
VOT_Field* field =
    stream->next()
    field->name()
    field->ID()
    field->ucd()
    field->ref()
    field->datatype()
    field->width()
    field->precision()
    field->unit()
```

MyRO: My Record Oriented privilege system

- A set of tools which provide row-level table access
  - A row-based grant mechanism for MySQL tables similar to that of a Unix file system. Table records are accessible depending on ownership and group membership. It means a user can access only those records it is allowed to "`select/update/delete`".
  - A consequence of this is that different users reading the same table will see different records.
- The software
  - The software components of MyRO are a Perl script used to perform administrative tasks, a C library and a set of MySQL functions. The process of protecting tables is completely transparent to the final user. After MyRO has been installed and configured by the database administrator, users can access the database without even know that MyRO is working.

MyRO: My Record Oriented privilege system

```
mysql> DESCRIBE Obslog_myro;
+----------+----------------------+------+-----+------------+
| Field    | Type                 | Null | Key | Default    |
+----------+----------------------+------+-----+------------+
| autoID   | int(10) unsigned     | NO   | PRI | NULL       |
| datein   | datetime             | NO   |     | 0000-00-00|
| htmID_6  | smallint(5) unsigned | NO   |     | 0          |
| ras      | char(12)             | NO   |     |            |
| decs     | char(12)             | NO   |     |            |
| radeg    | double unsigned      | NO   |     | 0          |
| decdeg   | double               | NO   |     | 0          |
| az       | double unsigned      | NO   |     | 0          |
| alt      | double               | NO   |     | 0          |
| telescop | char(20)             | NO   |     |            |
| instrume | char(8)              | NO   |     |            |
…
| pi_uname  | enum('nobody','user1','user2') | NO   |     | nobody     |
| my_uid    | tinyint(3) unsigned            | YES  |     | NULL       |
| my_gid    | tinyint(3) unsigned            | YES  |     | NULL       |
| my_perm   | tinyint(3) unsigned            | YES  |     | NULL       |
+-----------+--------------------------------+------+-----+------------+
```

Beijing, 11-07-2011

MyRO: My Record Oriented privilege system

```
mysql> show tables;
+----------------+
| Tables_in_myro |
+----------------+
| grp            |
| myro           |
| tbl            |
| usr            |
| usrgrp         |
+----------------+
mysql> describe usr;
+---------+----------------------+------+-----+---------+----------------+
| Field   | Type                 | Null | Key | Default | Extra          |
+---------+----------------------+------+-----+---------+----------------+
| uid     | tinyint(3) unsigned | NO   | PRI | NULL    | auto_increment |
| usr     | char(50)             | NO   | UNI | NULL    |                |
| su      | tinyint(1)           | NO   |     | 0       |                |
| defgid  | tinyint(3) unsigned | NO   |     | 1       |                |
| defperm | tinyint(3) unsigned | NO   |     | 11      |                |
| descr   | varchar(200)         | YES  |     | NULL    |                |
| email   | varchar(50)          | YES  |     | NULL    |                |
| flag    | tinyint(1)           | NO   |     | 1       |                |
+---------+----------------------+------+-----+---------+----------------+
```


Sky objects ⇒ 2-d tables indexing

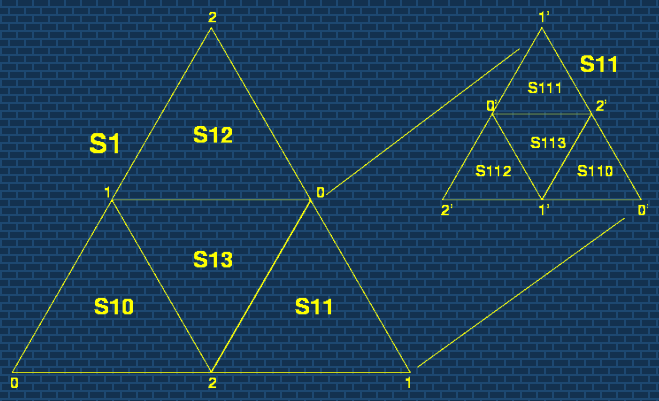
Possible indexing of a 2-d (3-d, ...) table:

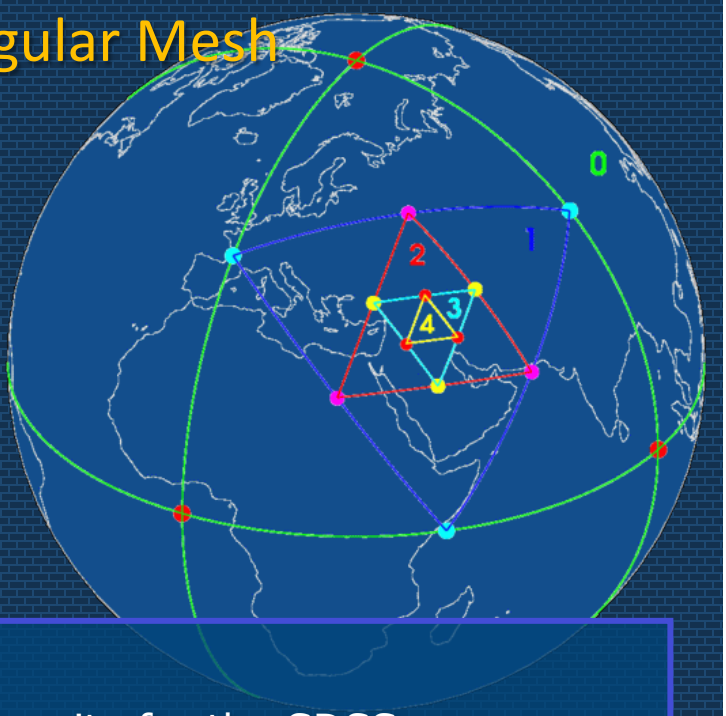
1. Split the sphere at your personal convenience – no index
2. Index (B-tree) on one single axis (e.g. declination)
3. Use an intrinsic (DB) spatial index like the R-tree
4. Use a function to map 2-d ⇔ 1-d then use B-tree

The most efficient is the last one, i.e. to have the possibility to allow the DB server to manage data on the sphere (2-d) exactly in the same way as it manages one dimensional data tables ⇒ using B-tree schema.

Various sphere coverage (tesselletion) functions have been proposed and used, but for Astronomical purposes the most common are *HTM* and *HEALPix* (excluding the geographic-like grids).

HTM: Hierarchical Triangular Mesh





HTM: `www.sdss.jhu.edu/htm/`

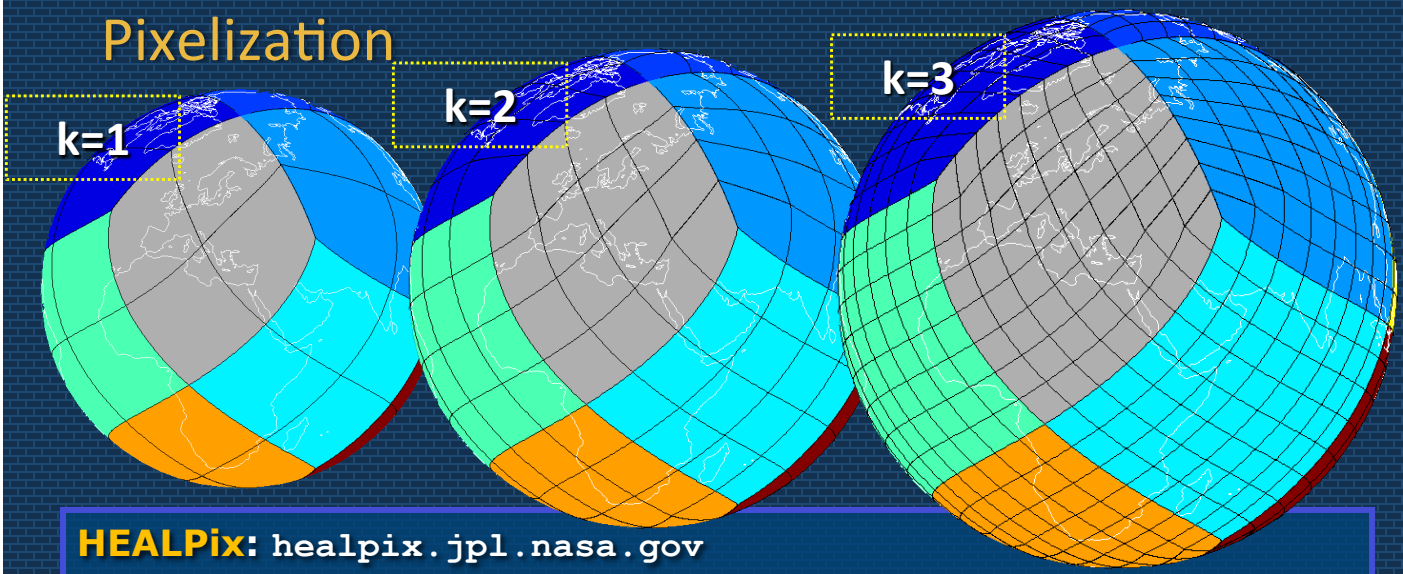
Invented at Johns Hopkins University for the SDSS survey.
The total Nr of pixels (*trixels*) in a map is set by *depth*:

$d \in [0 , 25]$ *(up to 30 possible!)* $N_{pix} = 8 \cdot 4^d$

ID_{range}: $[N_{pix} , 2 \cdot N_{pix} - 1]$

Max res.: $9.0 \cdot 10^{15}$ px → $7.7'' \cdot 10^{-3}$ (*24 cm on Earth*)

HEALPix: Hierarchical Equal Area isoLatitude Pixelization



HEALPix: `healpix.jpl.nasa.gov`

Invented at ESO for COBE. Also used by WMAP, Planck, etc.
The total Nr of pixels in a map is set by *order*: $k \in [0 , 29]$

$N_{side} = 2^k$ $N_{pix} = 12 \cdot N^2_{side}$

ID_{range}: $[0 , N_{pix} - 1]$ $\Omega_{pix} = \pi / (3 \times N^2_{side})$

Max res.: $3.46 \cdot 10^{18}$ pix → $3.93'' \cdot 10^{-4}$ (*1.2 cm on Earth*)

DIF: Dynamic Indexing Facility

(DIF indexed) **StarCat** table

...	RA	Decl	...	htmID_6

DIF added field+index

DIF.dif table

param	ID	full

Dynamically generated

JOIN
htmID_6 = ID

StarCat_htm VIEW

...	RA	Decl	...	htmID_6

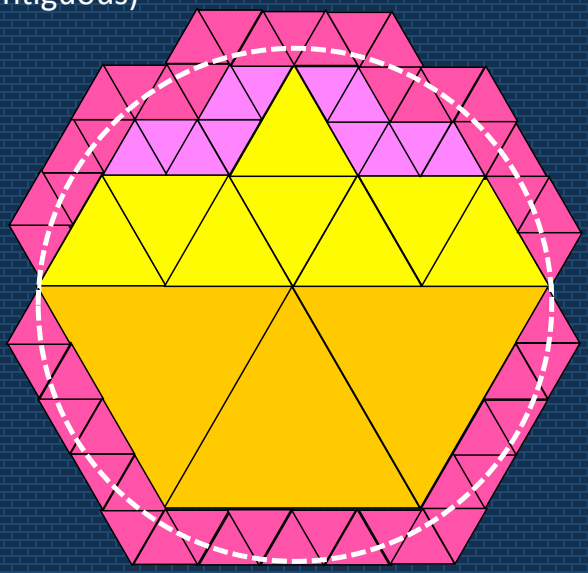
Example:
```
Select * From StarCat_htm Where DIF_Circle(82,12,6);
```

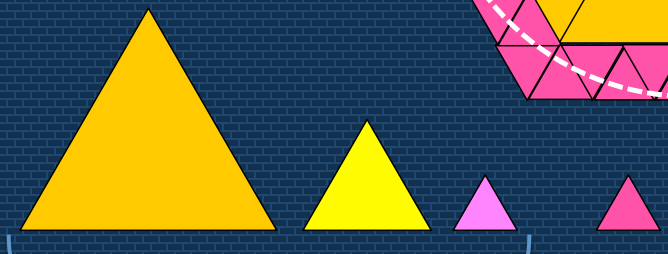
Center (°) & radius (′)

```
Select HEALPLookup(0,8,RA,Decl)
  From GSC23_htm Where DIF_Rect(20.1,30.5,8);
```


DIF: HTM multi-depth progressive erosion

Example with 3 depths
(don't need to be contiguous)





Full

Partial

DIF: MySQL extension

UDF Functions:

DIF.getHTMDepth	(db, tab)
DIF.getHEALPOrder	(db, tab)
DIF.getHEALPNested	(db, tab)
DIF.getRa	(db, tab)
DIF.getDec	(db, tab)
HTMBary	(d, ID)
HTMBaryC	(d, RA, Dec)
HTMBaryDist	(d, ID, RA, Dec)
HTMNeighb	(d, ID)
HTMNeighbC	(d, RA, Dec)
HEALPBary	(s, k, ID)
HEALPBaryC	(s, k, RA, Dec)
HEALPBaryDist	(s, k, ID, RA, Dec)
HEALPNeighb	(s, k, ID)
HEALPNeighbC	(s, k, RA, Dec)
Sphedist	(R1 ,D1, R2, D2)

HTMLookup	(d, RA, Dec)
HEALPLookup	(s, k, RA, Dec)

DIF engine Functions:

DIF_Circle	(RA, Dec, R)
DIF_Rect	(RA, Dec, S1, *S2*)
DIF_RectV	(RA1, Dec1, ...)
DIF_HTMNeighbC	(RA, Dec)
DIF_HEALPNeighbC	(RA, Dec)
DIF_setHTMDepth	(d)
DIF_setHEALPOrder	(s, k)
DIF_FineSearch	(*var*)
DIF_Sphedist	(RA1, Dec1, RA2, Dec2)

```
Select HEALPLookup(0,8,RA,Dec)
From V Where DIF_Rect(0,0,20);
```

HTM and HEALPix facts

Typically used pixelization resolutions:

– HTM: 6, 8, 10, 12 – HEALPix: 6, 8, 10, 12

	d / k	Npix	<Area>	Obj/Pix	Bytes
HTM	6	32,768	1.26 deg^2	89,290	2
	8	524,288	283 arcmin^2	5,580	3
	10	8,388,608	18 arcmin^2	348	3
	12	134,217,728	1 arcmin^2	21	4
HEALPix	6	49,152	0.84 deg^2	59,526	2
	8	786,432	189 arcmin^2	3,720	3
	10	12,582,912	12 arcmin^2	232	3
	12	201,326,592	0.7 arcmin^2	14	4

For generic use, ~1000 Obj/Pix gives best performance