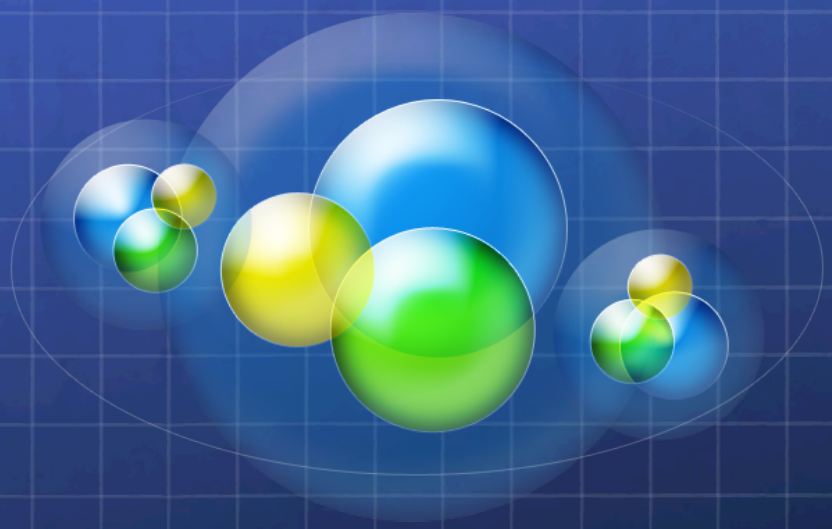


Relational databases

in ASTRONOMY

Luciano Nicastro – IASF-INAF, Bologna

Relational database principles

- A database (DB) is a very large collection of related data, usually too large to fit into main memory and often used by many users simultaneously.
- A database system implements real-world concepts and applications (entities, properties, relationships).
- The invention of this database system has standardized the way data is stored and processed.
- Most of the database systems in use today are based on the relational system. *However other types of DBs exist! E.g. OODBs*.
- A relational database consists of a collection of tables that store particular sets of data among which relationships can exist.
- Application in all types of data management: banks, airlines and travel services, www applications, e-commerce, telecommunication, digital libraries, scientific data, etc.

Where to look

- en.wikipedia.org/wiki/Relational_database
- www.mysql.com / dev.mysql.com
- www.postgresql.org
- www.sqlite.org

www.stellarium.com

Tabular view of the data

Airline case (three tables):

Flight

flightNo	depart	destination	miles
101	BOS	LAX	3000
102	PVD	LAX	2900
…	…	…	…

Passenger

pName	ffNumber	DoB	milesEarned
Joe	1001	2010	12000
Mary	1002	2011	11000
…	…	…	…

FlewIn

flightNo	ffNumber	date	year
101	1001	Jan 4	2010
102	1002	Jan 5	2011
…	…	…	…

Messier catalogue

⋮

2

1

The Messier catalogue example

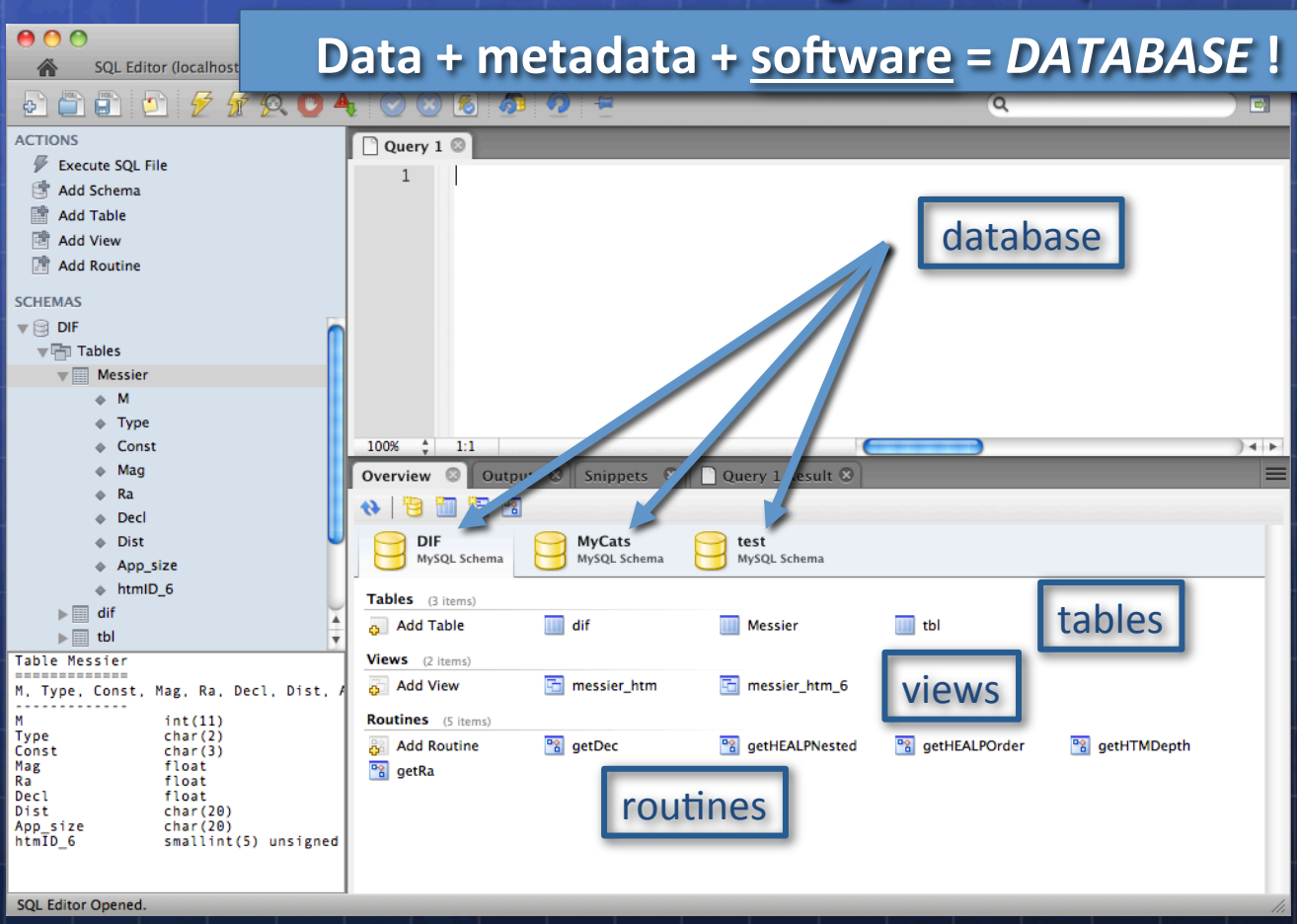
Metadata

Id	Type	Const	Mag	Ra_h	Ra_m	Dec_d	Dec_p	Dist	App_size
int	string	string	float	float	float	float	float	string	string
1	BN	Tau	8.2	05	34.5	+22	01.0	6.3 kly	6'x4'
2	GC	Aqu	6.3	21	33.5	00	-49.0	36.2 kly	12.9'
3	GC	CVn	6.3	13	42.2	+28	23.0	30.6 kly	16.2'
4	GC	Sco	6.4	16	23.5	-26	31.5	6.8 kly	26.3'
5	GC	Ser	6.2	15	18.6	+02	05.0	22.8 kly	17.4'
6	OC	Sco	4.2	17	40.4	-32	13.8	2 kly	33'
7	OC	Sco	4.1	17	53.9	-34	47.0	800 ly	80.0'
8	BN	Sag	6.0	18	04.1	-24	18.0	5200 ly	90'x40'
9	GC	Oph	7.3	17	19.2	-18	31.0	26.4 kly	9.3'

- Data + metadata = file FITS, XML, VOTable, etc.
- Data + metadata + software = *DATABASE* !

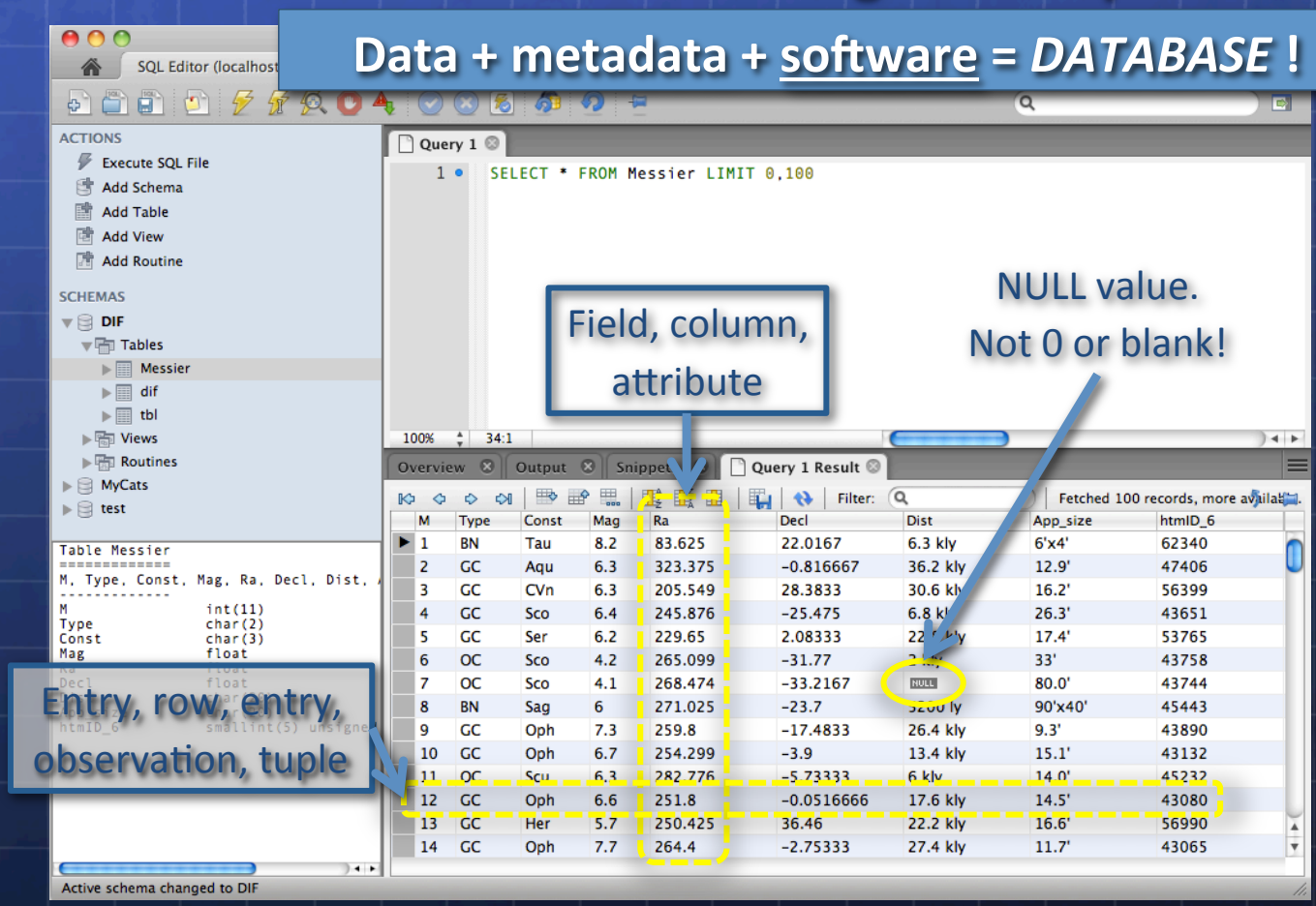
DBMS: the Messier catalogue example

Data + metadata + software = *DATABASE* !



DBMS: the Messier catalogue example

Data + metadata + software = *DATABASE* !



DBMS: the Messier table structure

Must know what type of query can be performed on the fields.

Data Type

Index?

Default value

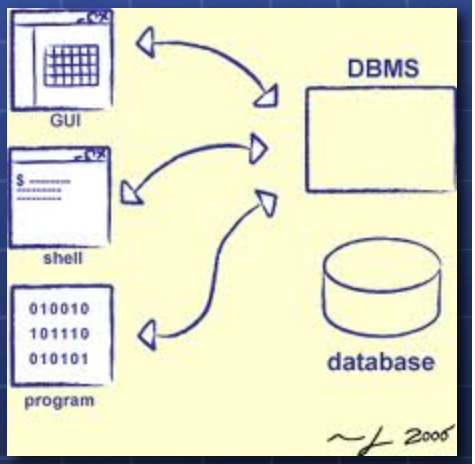
```
mysql> describe Messier;
+----------+-----------------------+------+-----+---------+-------+
| Field    | Type                  | Null | Key | Default | Extra |
+----------+-----------------------+------+-----+---------+-------+
| M        | int(11)               | NO   | PRI | NULL    |       |
| Type     | char(2)               | YES  |     | **      |       |
| Const    | char(3)               | YES  |     | ***     |       |
| Mag      | float                 | YES  |     | NULL    |       |
| Ra       | float                 | YES  |     | NULL    |       |
| Decl     | float                 | YES  |     | NULL    |       |
| Dist     | char(20)              | YES  |     | NULL    |       |
| App_size | char(20)              | YES  |     | unknown |       |
| htmID_6  | smallint(5) unsigned  | NO   | MUL | 0       |       |
+----------+-----------------------+------+-----+---------+-------+
9 rows in set (0.00 sec)

mysql>
```

Primary Key

Foreign Key

How to query a database



Server based DBs require connection and authentication; then you can start querying using the SQL.

- Via dedicated GUI interface.
  - Also allows DB management.
- Via terminal.
  - Type your query (command) at the prompt and see the output.
- Via custom applications using API libraries.
  - Transfer data from/to your software program.
  - ODBC drivers → translation layer between applications and DBMS.
  - Can use (almost) any programming language to query a DBMS.

How to query a database

Via terminal.

nicastro@Mac /Users/nicastro/Utils — mysql — 82×20

```
mysql> select * from Messier where M < 12;
+----+------+-------+------+----------+------------+----------+----------+---------+
| M  | Type | Const | Mag  | Ra       | Decl       | Dist     | App_size | htmID_6 |
+----+------+-------+------+----------+------------+----------+----------+---------+
|  1 | BN   | Tau   |  8.2 |   83.625 |    22.0167 | 6.3 kly  | 6'x4'    |   62340 |
|  2 | GC   | Aqu   |  6.3 |  323.375 |  -0.816667 | 36.2 kly | 12.9'    |   47406 |
|  3 | GC   | CVn   |  6.3 |  205.549 |    28.3833 | 30.6 kly | 16.2'    |   56399 |
|  4 | GC   | Sco   |  6.4 |  245.876 |   -25.475  | 6.8 kly  | 26.3'    |   43651 |
|  5 | GC   | Ser   |  6.2 |   229.65 |    2.08333 | 22.8 kly | 17.4'    |   53765 |
|  6 | OC   | Sco   |  4.2 |  265.099 |     -31.77 | 2 kly    | 33'      |   43758 |
|  7 | OC   | Sco   |  4.1 |  268.474 |   -33.2167 | NULL     | 80.0'    |   43744 |
|  8 | BN   | Sag   |    6 |  271.025 |      -23.7 | 5200 ly  | 90'x40'  |   45443 |
|  9 | GC   | Oph   |  7.3 |    259.8 |   -17.4833 | 26.4 kly | 9.3'     |   43890 |
| 10 | GC   | Oph   |  6.7 |  254.299 |       -3.9 | 13.4 kly | 15.1'    |   43132 |
| 11 | OC   | Scu   |  6.3 |  282.776 |   -5.73333 | 6 kly    | 14.0'    |   45232 |
+----+------+-------+------+----------+------------+----------+----------+---------+
11 rows in set (0.00 sec)

mysql>
```

How to query a database

Via custom applications using API libraries. Here IDL (custom).

```
@mcs_usrlib    ; User contributed Functions and Pro

PRO simpleqry

; This is the common section for the "mcs_usrlib"
COMMON DBinfo,  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
;--

; Required info
  c_user  = 'generic'  &  c_password = 'password'  &  c_dbname = 'DIF'
  c_query = 'SELECT * FROM Messier WHERE M < 10'

; All in one func.: open connection, execute query, show the result, close
  status = DBExec_ShowQry(/use_saved_info) ; return 0 if OK

  IF (status NE 0) THEN RETURN  ; Error?

; The result set is stored in a structure
  PRINT, 'Info on the returned structure:'
  HELP, /STRU, tab_stru

END
```

How to query a database

Via WEB (interface): observations log

http://ross.iasfbo.inaf.it/REMDB/obslog.php

Select ALL where ImgNight ≥ 2010-05-07 ≤ 2010-05-07 Distinct Obj-PI-Filt. Retrieve imgs

Order by Date Ascending Format LongTab Coords STD string 20 imgs/pg Go! Log-out ross

?	Filename	Object	RA	Dec	uniqueID	Date	Time	PI-CoI	Texp	Filter	ObsType	MJD
	V567_Sco_1_1_K_5	V567_Sco	16:49:50.64	-40:55:05.5	261430105	2010-05-08	10:20:09	Marcio_Catelan	1	K	GENSTAR	55324.43066
	V567_Sco_1_1_K_4	V567_Sco	16:49:50.64	-40:55:05.5	261430104	2010-05-08	10:20:02	Marcio_Catelan	1	K	GENSTAR	55324.43058
	V567_Sco_1_1_K	V567_Sco	16:49:50.64	-40:55:05.5	261430199	2010-05-08	10:19:56	Marcio_Catelan	1	K	GENSTAR	55324.43051
	V567_Sco_1_1_K_sky	V567_Sco	16:49:50.64	-40:55:05.5	261430198	2010-05-08	10:19:56	Marcio_Catelan	1	K	GENSTAR	55324.43051
	V567_Sco_1_1_K_3	V567_Sco	16:49:50.64	-40:55:05.5	261430103	2010-05-08	10:19:56	Marcio_Catelan	1	K	GENSTAR	55324.43051
	V567_Sco_1_1_K_2	V567_Sco	16:49:50.64	-40:55:05.5	261430102	2010-05-08	10:19:49	Marcio_Catelan	1	K	GENSTAR	55324.43043
	V567_Sco_1_1_K_1	V567_Sco	16:49:50.64	-40:55:05.5	261430101	2010-05-08	10:19:42	Marcio_Catelan	1	K	GENSTAR	55324.43035
	WY_Sco_2_10_K_5	WY_Sco	16:33:20.40	-26:11:14.6	261420105	2010-05-08	10:17:24	Marcio_Catelan	10	K	GENSTAR	55324.42875
	WY_Sco_2_10_K_4	WY_Sco	16:33:20.40	-26:11:14.6	261420104	2010-05-08	10:17:08	Marcio_Catelan	10	K	GENSTAR	55324.42856
	WY_Sco_2_10_K	WY_Sco	16:33:20.40	-26:11:14.6	261420199	2010-05-08	10:16:52	Marcio_Catelan	10	K	GENSTAR	55324.42838
	WY_Sco_2_10_K_sky	WY_Sco	16:33:20.40	-26:11:14.6	261420198	2010-05-08	10:16:52	Marcio_Catelan	10	K	GENSTAR	55324.42838
	WY_Sco_2_10_K_3	WY_Sco	16:33:20.40	-26:11:14.6	261420103	2010-05-08	10:16:52	Marcio_Catelan	10	K	GENSTAR	55324.42838
	WY_Sco_2_10_K_2	WY_Sco	16:33:20.40	-26:11:14.6	261420102	2010-05-08	10:16:36	Marcio_Catelan	10	K	GENSTAR	55324.42819
	WY_Sco_2_10_K_1	WY_Sco	16:33:20.40	-26:11:14.6	261420101	2010-05-08	10:16:21	Marcio_Catelan	10	K	GENSTAR	55324.42802
	GK_Nor_2_12_K_5	GK_Nor	15:34:50.88	-58:23:58.6	261410105	2010-05-08	10:13:46	Ronald_Mennickent	12	K	GENSTAR	55324.42623
	GK_Nor_2_12_K_4	GK_Nor	15:34:50.88	-58:23:58.6	261410104	2010-05-08	10:13:28	Ronald_Mennickent	12	K	GENSTAR	55324.42602
	IMG126I240	GK_Nor	15:34:44.64	-58:24:34.6	261410200	2010-05-08	10:13:12	Ronald_Mennickent	30	I	GENSTAR	55324.42584
	GK_Nor_2_12_K	GK_Nor	15:34:50.88	-58:23:58.6	261410199	2010-05-08	10:13:10	Ronald_Mennickent	12	K	GENSTAR	55324.42581
	GK_Nor_2_12_K_sky	GK_Nor	15:34:50.88	-58:23:58.6	261410198	2010-05-08	10:13:10	Ronald_Mennickent	12	K	GENSTAR	55324.42581
	GK_Nor_2_12_K_3	GK_Nor	15:34:50.88	-58:23:58.6	261410103	2010-05-08	10:13:10	Ronald_Mennickent	12	K	GENSTAR	55324.42581

Selected **1817** entries, shown 1 - 20 UT now: Jun 07 2010 09:42:48

Page: 1 2 3 4 >> 14 ... 91 1

Used query:
SELECT * FROM REMImgs.Obslog WHERE (date_obs > '2010-05-07q 22:00:00' AND date_obs < '2010-05-08 12:00:00') ORDER BY date_obs DESC LIMIT 0,20

Your **WHERE** *clause:* ? Distinct... Retrieve imgs Go!

REM images browser:
Read the usage permission rules.

Credits

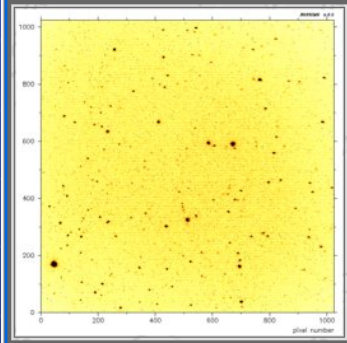
Written by L. Nicastro
Version: *0.2e, 17-Jun-2008*

Go ***BACK*** or to the **REM images browser page**

How to query a database

Select ALL where ImgNight ≥ 2010-05-07 ≤ 2010-05-07 Distinct Obj-PI-Filt. Retrieve imgs

Order by Date Ascending Format LongTab Coords STD string 5 imgs/pg Go! Log-out ross



317787. IMG126I240 (SExCat): Tools RefCats Aladin 17

Object: **GK_Nor** (GENSTAR) PI: **Ronald_Mennickent**

RA: **15:34:44.64** Dec: **-58:24:34.6** Date: **2010-05-08** UT: **10:13:12**

Gl: **322.973** Gb: **-2.042** MJD: **55324.42584**

Filter: **I** Exp.: **30** s

ObsID: **26141** SubID: **2** DithID: **0**

ObID: **112897** PropID: **21701**

Processing status ID: 3

ROSS Raw Direct image

Sky chart: GSC-2.3, USNO-B1.0, 2MASS, *Google Sky*

Inserted on *2010-05-08 20:26:08*. PI user name: **rmennickent**

?	Filename	Object	RA	Dec	uniqueID	Date	Time	PI-CoI	Texp	Filter	ObsType	MJD
	GK_Nor_2_12_K_4	GK_Nor	15:34:50.88	-58:23:58.6	261410104	2010-05-08	10:13:28	Ronald_Mennickent	12	K	GENSTAR	55324.42602
	IMG126I240	GK_Nor	15:34:44.64	-58:24:34.6	261410200	2010-05-08	10:13:12	Ronald_Mennickent	30	I	GENSTAR	55324.42584
	GK_Nor_2_12_K	GK_Nor	15:34:50.88	-58:23:58.6	261410199	2010-05-08	10:13:10	Ronald_Mennickent	12	K	GENSTAR	55324.42581
	GK_Nor_2_12_K_sky	GK_Nor	15:34:50.88	-58:23:58.6	261410198	2010-05-08	10:13:10	Ronald_Mennickent	12	K	GENSTAR	55324.42581
	GK_Nor_2_12_K_3	GK_Nor	15:34:50.88	-58:23:58.6	261410103	2010-05-08	10:13:10	Ronald_Mennickent	12	K	GENSTAR	55324.42581

Selected **1817** entries, shown 16 - 20 UT now: Jun 07 2010 09:52:59

Page: 1 2 3 4 5 6 7 >> 17 ... 364 4

Used query:
SELECT * FROM REMImgs.Obslog WHERE (date_obs > '2010-05-07q 22:00:00' AND date_obs < '2010-05-08 12:00:00') ORDER BY date_obs DESC LIMIT 15,5

Your **WHERE** *clause:* ? Distinct... Retrieve imgs Go!

REM images browser:
Read the usage permission rules.

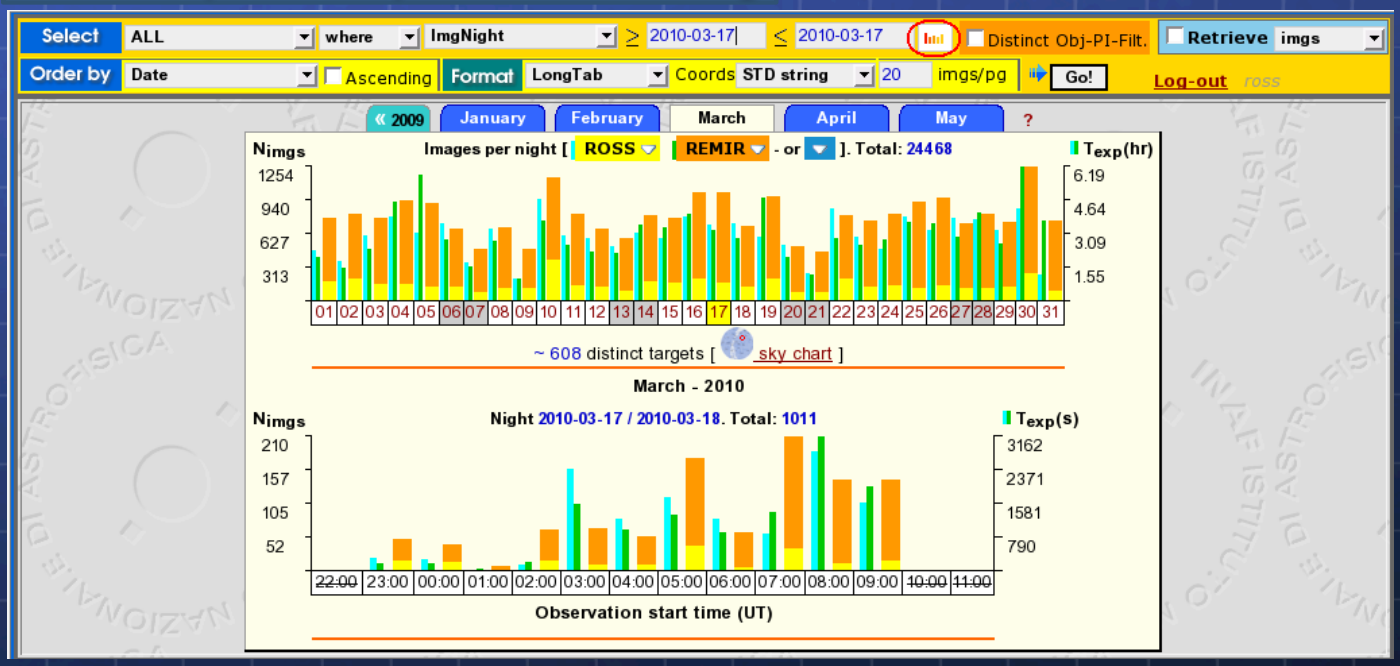
Credits

Written by L. Nicastro
Version: *0.2e, 17-Jun-2008*

Go ***BACK*** or to the **REM images browser page**

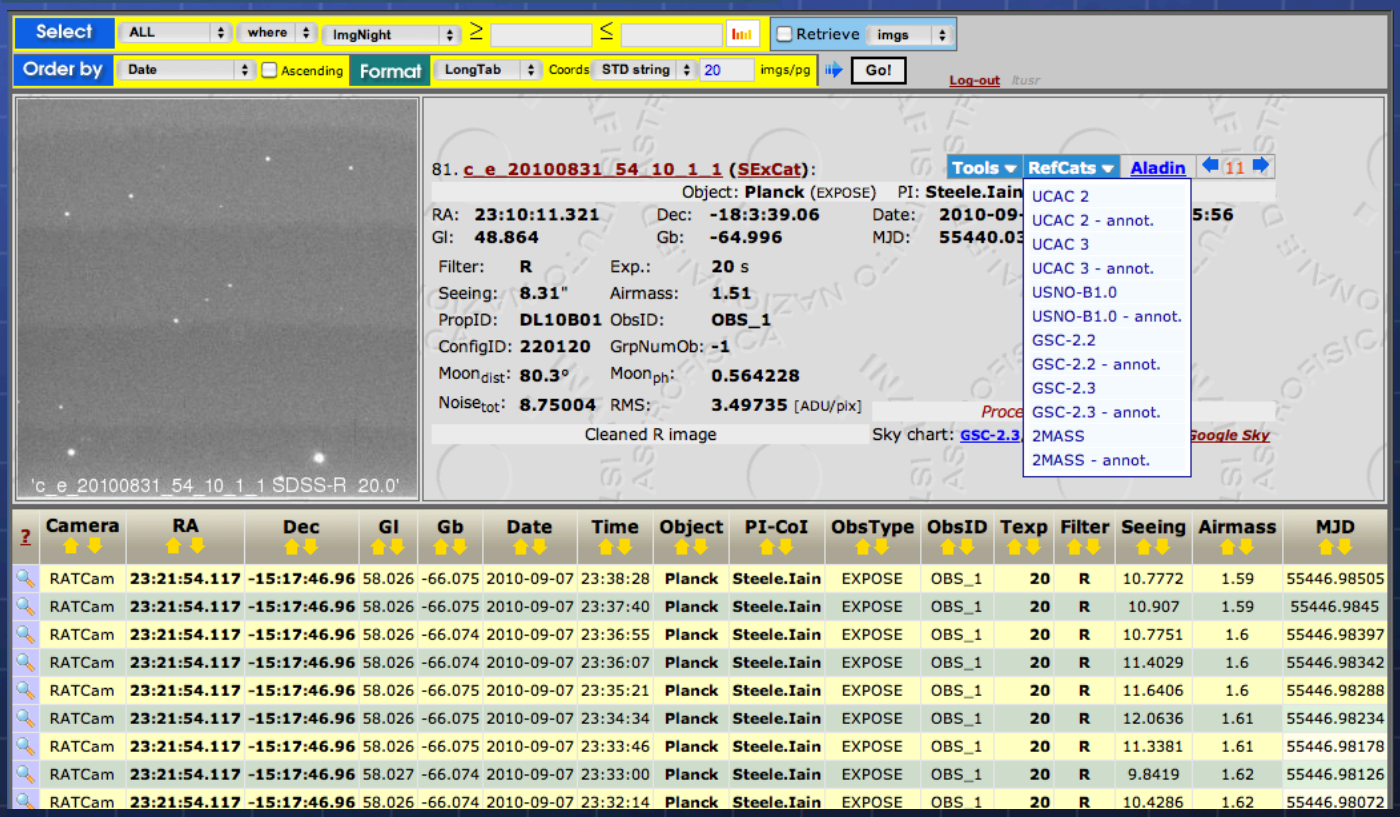
How to query a database

Via WEB (interface): observations log



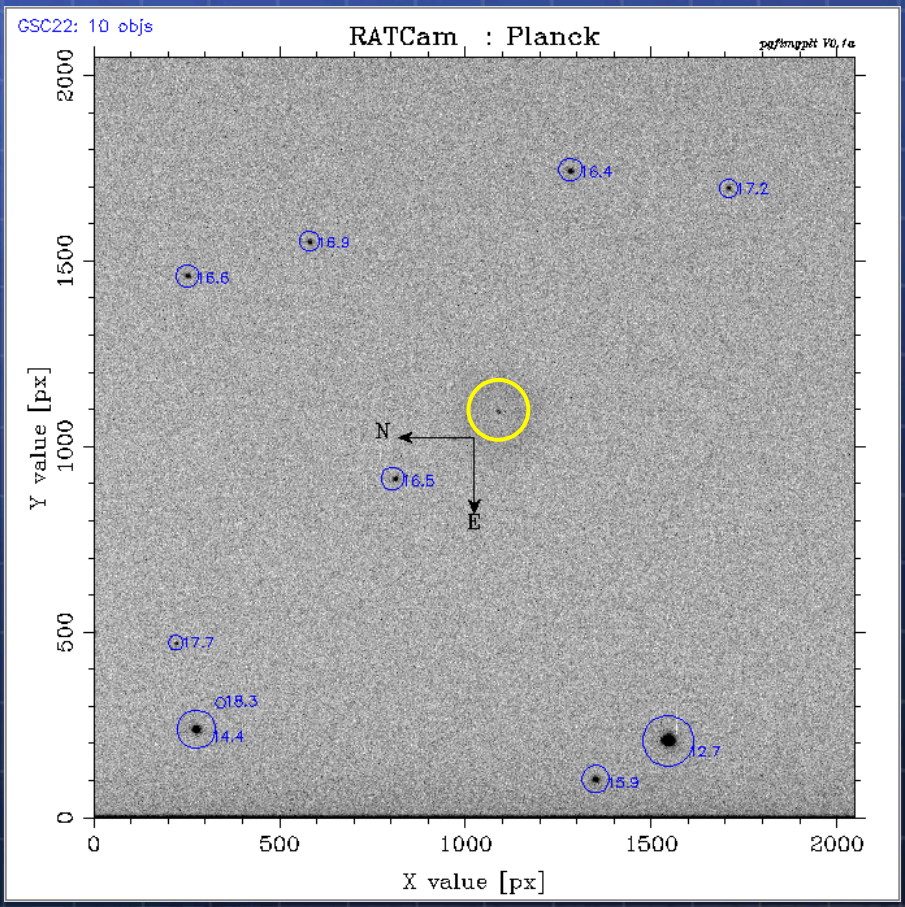
How to query a database

Via WEB (interface): observations log



?	Camera	RA	Dec	Gl	Gb	Date	Time	Object	PI-CoI	ObsType	ObsID	Texp	Filter	Seeing	Airmass	MJD
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.075	2010-09-07	23:38:28	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	10.7772	1.59	55446.98505
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.075	2010-09-07	23:37:40	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	10.907	1.59	55446.9845
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.074	2010-09-07	23:36:55	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	10.7751	1.6	55446.98397
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.074	2010-09-07	23:36:07	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	11.4029	1.6	55446.98342
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.075	2010-09-07	23:35:21	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	11.6406	1.6	55446.98288
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.075	2010-09-07	23:34:34	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	12.0636	1.61	55446.98234
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.075	2010-09-07	23:33:46	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	11.3381	1.61	55446.98178
	RATCam	23:21:54.117	-15:17:46.96	58.027	-66.074	2010-09-07	23:33:00	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	9.8419	1.62	55446.98126
	RATCam	23:21:54.117	-15:17:46.96	58.026	-66.074	2010-09-07	23:32:14	Planck	Steele.Iain	EXPOSE	OBS_1	20	R	10.4286	1.62	55446.98072

How to query a database

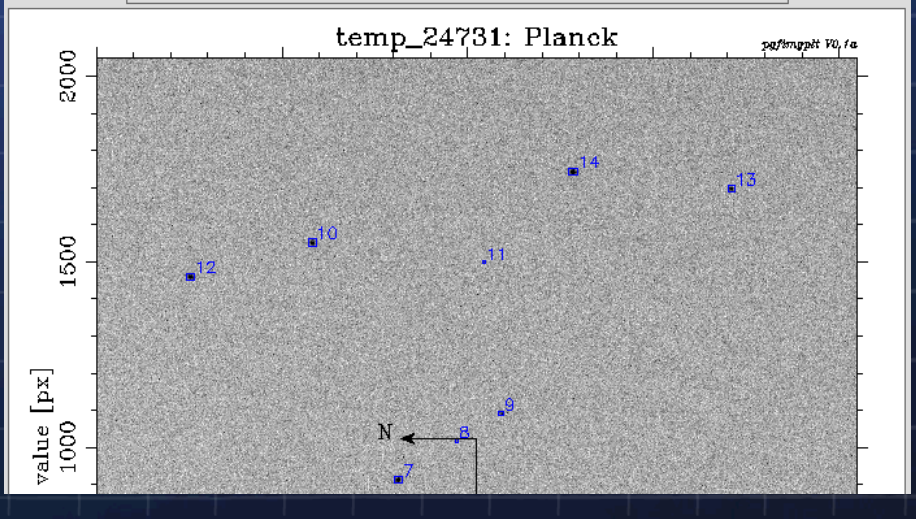


How to query a database

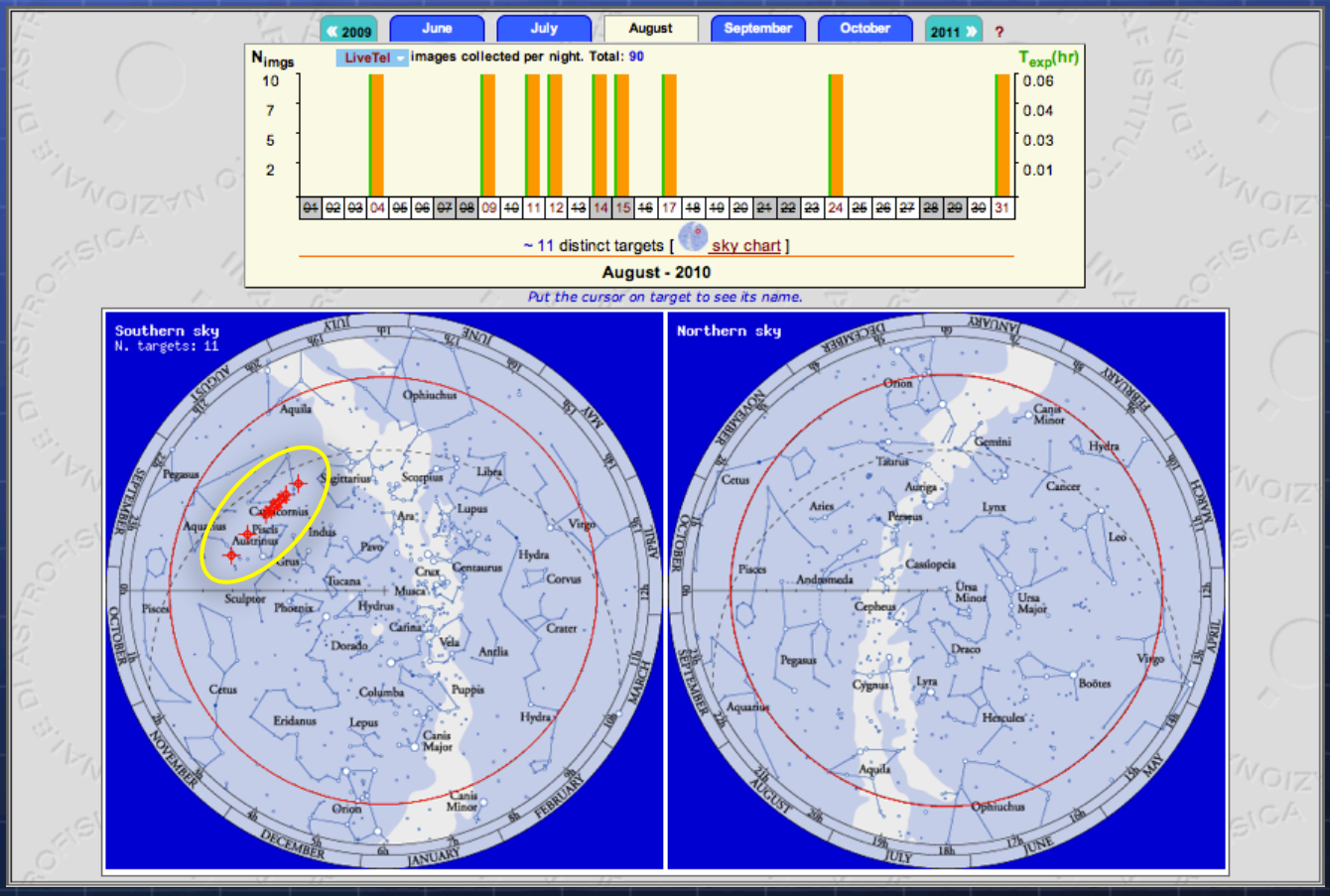
?

ID	Xmin	Xmax	X	Ymin	Ymax	Y	MagAper	ErrMagA	MagBest	ErrMagB	Bckg	Elong	Flags
1	1	2048	416.823	1	14	1.229	13.0359	0.0207	6.5977	0.0014	10.5106	266.659	27
2	801	2048	1486.860	1	4	1.000	13.1887	0.0223	7.2698	0.0015	9.939198	378.787	27
3	1338	1364	1350.601	94	114	103.900	12.6281	0.0172	10.7004	0.0105	9.681357	1.154	0
4	1521	1572	1547.138	186	235	208.994	9.5611	0.0038	7.6618	0.0017	9.334565	1.075	0
5	257	292	276.028	223	255	238.210	11.3088	0.0087	9.4202	0.0043	9.124107	1.105	0
6	216	231	222.694	465	477	470.315	14.3995	0.0515	12.6550	0.0394	8.255416	1.193	0
7	802	824	812.485	904	923	913.301	13.2921	0.0250	11.4628	0.0165	8.551344	1.078	0
8	967	975	971.231	1014	1025	1020.227	15.1114	0.0878	13.6103	0.0810	8.493636	1.319	0
9	1082	1097	1089.879	1088	1099	1093.653	14.4816	0.0546	12.5177	0.0404	8.331888	1.174	2
10	572	591	581.820	1543	1561	1551.849	13.5771	0.0299	11.6800	0.0201	8.701831	1.156	0
11	1039	1048	1042.413	1495	1504	1499.328	15.5923	0.1288	13.8430	0.1132	8.718512	2.199	2
12	241	264	253.852	1452	1469	1460.155	13.4092	0.0269	11.5108	0.0192	8.320092	1.184	0
13	1701	1718	1709.561	1689	1705	1696.421	14.0045	0.0393	12.0857	0.0280	8.810527	1.146	0
14	1273	1295	1283.316	1733	1753	1742.609	13.2487	0.0244	11.3234	0.0163	8.871246	1.126	0

Median Width and Height (pix.): 21.50 18.50 Average Width and Height (pix.): 22.83 20.50



How to query a database



How to query a database

Via WEB (interface): catalogues browsing

Format: LongTab | Coords STD string | 20 objs/pg | Gl/Gb | Sep | Sky-chart | Show | New Query

?	RA	Dec	Sep (')	htmID	gscID	Epoch	RApm	DECpm	Fmag	Jmag	Vmag	Nmag	J-F	F-N	Class	Ext	Ecc	PosAng
1	01 59 50.5	-00 00 51	2.53	33382	5034	1986.68	0	0	19.349	-	-	-	-	-	0	2.22	0.19	156
2	01 59 50.7	-00 01 23	2.7	33382	4989	1986.68	0	0	19.44	21.537	-	18.152	2	1.2	0	1.88	0.07	57
3	01 59 51.2	+00 01 01	2.43	63897	5662	1996.78	-1.7	-20.5	20.281	22.129	-	-	1.8	-	3	1.72	0.37	135
4	01 59 51.3	+00 01 14	2.51	63897	7406	1996.78	0	0	19.87	21.976	-	-	2.1	-	3	1.95	0.13	41
5	01 59 51.5	-00 01 27	2.57	33382	4980	1986.68	0	0	19.739	-	-	-	-	-	3	2.1	0.21	57
6	01 59 51.9	+00 01 35	2.57	63897	8840	1996.78	0	0	20.344	-	-	-	-	-	3	1.56	0.36	178
7	01 59 51.1	-00 00 12	2.02	33382	9130	1996.78	0	0	20.48	-	-	-	-	-	3	1.54	0.42	160
8	01 59 52.2	+00 01 28	2.43	63897	373	1986.68	5.5	-12.6	16.213	18.332	-	15.146	2.1	1	0	3.1	0.07	175
9	01 59 53.0	-00 00 31	1.81	33382	5078	1986.68	0	0	15.496	17.704	-	14.873	2.2	0.6	0	3.68	0.08	103
10	01 59 53.1	+00 00 43	1.86	63897	292	1986.68	3.1	-18.7	15.914	16.804	-	15.755	0.8	0.1	0	3.22	0.01	162
11	01 59 53.7	-00 01 42	2.32	33382	6316	1984.6	0	0	-	22.134	-	-	-	-	3	1.1	0.37	46
12	01 59 54.3	+00 01 33	2.11	63897	8839	1996.78	0	0	20.457	-	-	-	-	-	3	1.43	0.37	108
13	01 59 54.4	-00 01 43	2.22	33382	9064	1996.78	0	0	20.517	-	-	-	-	-	3	1.53	0.43	160
14	01 59 55.5	+00 02 10	2.43	63897	8869	1996.78	0	0	20.193	-	-	-	-	-	3	2	0.35	95
15	01 59 56.4	+00 02 29	2.65	63897	500	1986.68	0	0	19.142	22.119	-	18.825	2.9	0.3	3	2.78	0.26	129
16	01 59 56.4	+00 00 56	1.3	63897	8817	1996.78	0	0	20.034	-	-	-	-	-	3	2.34	0.45	139
17	01 59 56.6	+00 01 33	1.76	63897	8838	1996.78	0	0	20.497	-	-	-	-	-	3	1.48	0.38	107
18	01 59 56.9	+00 00 31	0.93	63897	277	1986.68	-57.3	-117.7	19.943	22.141	-	-	2.1	-	3	2.11	0.38	89
19	01 59 57.2	+00 00 02	0.69	63897	232	1986.68	14.7	10	18.731	21.519	-	18.659	2.7	0	3	2.57	0.07	53
20	01 59 57.1	+00 00 52	1.01	63897	303	1986.68	-2.7	5.2	18.962	20.504	-	18.54	1.5	0.4	3	2.66	0.24	21

Selected **58** objects within 5 arcmin (shown 1 - 20) around RA: 02 00 00.0 Dec: 00 00 00 (Gl: 157.005220 Gb: -58.261640)

Page: 1 2 3 1

Used query:
SELECT * FROM GSC23 WHERE((htmID_6=33382 OR htmID_6=63897) AND (RAcs>10785000 AND RAcs<10815000) AND (DECcs>-14999 AND DECcs<15000)) ORDER BY racs ASC LIMIT 0,20

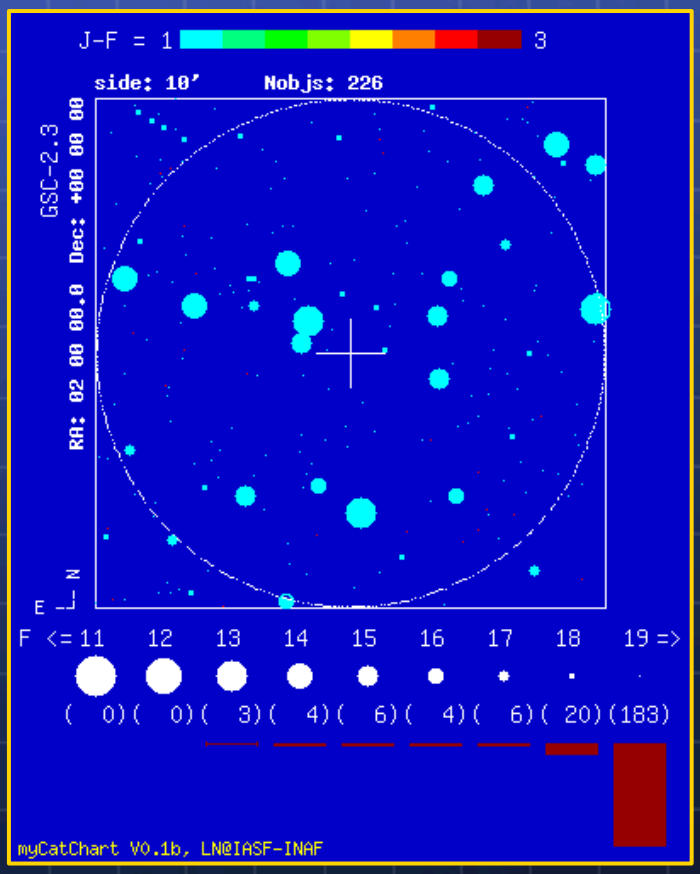
ROSS MySQL catalogues:
Read the usage permission rules

Credits

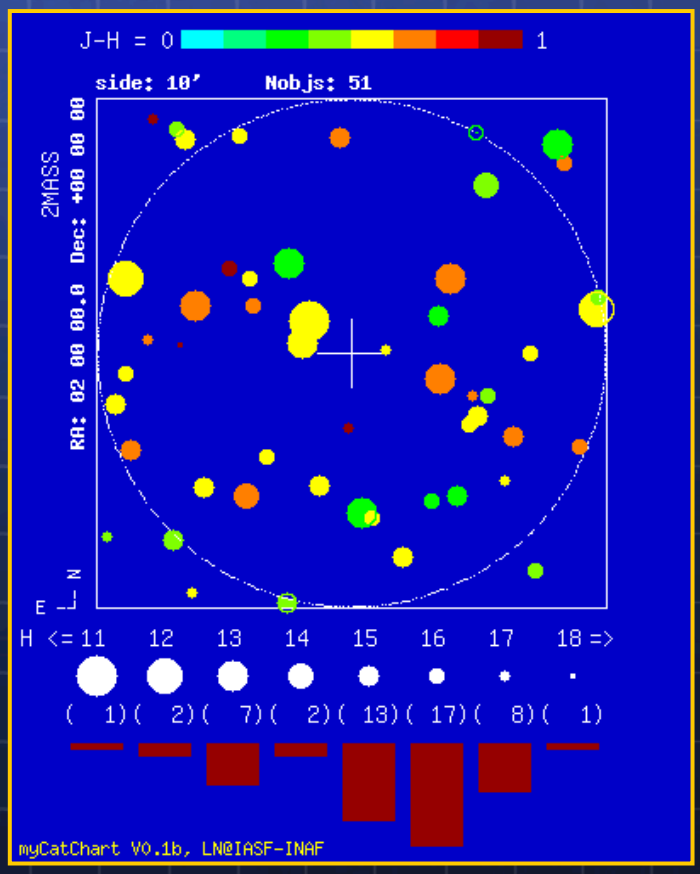
Written by L. Nicastro
Version: *0.5, 10-Jan-2005*

How to query a database

GSC2.3



2MASS



Tabular view of the data (again)

Airline case (three tables):

Flight

flightNo	depart	destination	miles
101	BOS	LAX	3000
102	PVD	LAX	2900
…	…	…	…

Passenger

pName	ffNumber	DoB	milesEarned
Joe	1001	2010	12000
Mary	1002	2011	11000
…	…	…	…

FlewIn

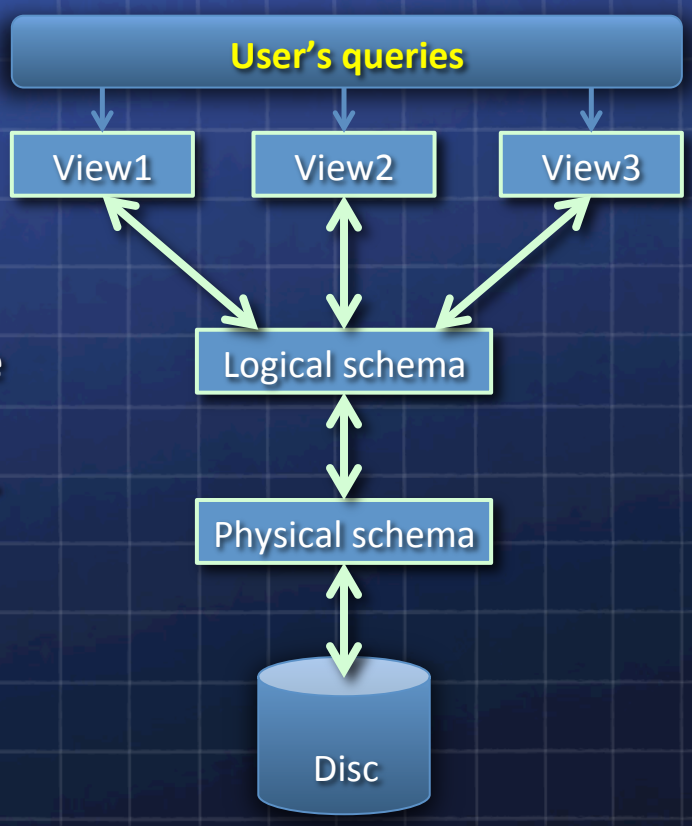
flightNo	ffNumber	date	year
101	1001	Jan 4	2010
102	1002	Jan 5	2011
…	…	…	…

Relational database principles

- The concept of a relational database derives from the principles of relational algebra, realized as a whole by the father of relational databases, E. F. Codd (1970 - *A Relational Model of Data for Large Shared Data Banks*).
- This theory established that "data should be independent of any hardware or storage system, and provided for automatic navigation between the data elements". In practice, this meant that data should be stored in tables (datasets) and that relationships could/ would exist between these different datasets.
- In practice, a nowadays relational database is implemented into a software package that serves a number of users connecting and performing queries via internet. In other words it implements a client-server system where the server can also be "distributed". We can refer to it as a Database Management System (DBMS or RDBMS).
- Interaction with a DBMS is performed via queries written using the Structured Query Language (SQL).

Level of abstraction

- External schema (view) describes how users see the data. Ex. *NoOfPassengers* → flightNo, date, numPassengers.
- Logical schema describes the logical structures used. Ex. Flight, Passenger, FlewIn.
- Physical schema describes files and indices. Ex. Index on flightNo for Flight and FlewIn.



Database Management System (DBMS): principles

A *DBMS* is a software package designed to store, manage and give access to databases.

DBMS compared to a simple file I/O:

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

We'll discuss these items in more detail.

Developer: why learn using DBMS

From the developer point of view, in addition to the advantages listed before:

- Shift from *computation* to *information*.
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project, astronomical datasets.
  - ... need for DBMS exploding.
- DBMS encompasses most of Computer Science, like:
  - OS, languages, theory, multimedia, logic, etc.

Developer: why use a DBMS

Let's list again the main advantages:

- Reduced application development time.
- Data independence.
- Efficient data access.
- Data integrity under updates.
- Concurrent access.
- Recovery from crashes.
- Security.
- Uniform data administration.

DBMS: reduced application development time

- Higher level of data abstraction.
- Application portability.
- Queries are written in a high level language tailored for database applications.

```
SELECT pname
FROM Passengers
WHERE flightNo = 101
```

What to get
From which dataset
Applying which filtering

In terms of relational algebra it is:

$\pi_L (\sigma_C (R))$ → SELECT L FROM R WHERE C

DBMS: data independence and efficient access

One of the most important benefits of using a DBMS!

- Applications can ignore how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.
- Efficient access.
  - Indices allow you to see only the "necessary" portion of data, as opposed to sequential access in files.

DBMS: performance & data integrity

- Concurrent access: DBMS ensures data is consistent.
  - e.g.: multiple airline staff trying to reserve a seat for different customers.
  - Users can pretend they are using a single-user system.
- Ideas.
  - Transactions: grouping multiple instructions (reads/writes) into one atomic unit.
  - Locks: locking of resources (tables).
- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently.

DBMS: crashes recovery, security, etc.

- If the system crashes in the middle of a transaction, recovery should be possible.
  - Ideas: logging, commit/rollback of transactions.
- DBMS also implement other features to facilitate administration, such as:
  - backup, replication and duplication, security, access control, privileges etc.
- Higher data availability can be obtained using distributed DBs.
  - Different architectures can be adopted.
  - More complex and requiring more "resources".

DBMS: transactions - atomicity

- Key concept is *transaction* (Xact), which is an *atomic* sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a *consistent state* if DB is consistent when the transaction begins. Utilize locking of resources and other protocols for guaranteeing consistency.
- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.
- Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of Xacts (it uses write ahead log – WAL).

DBMS: terminology

Data Models:

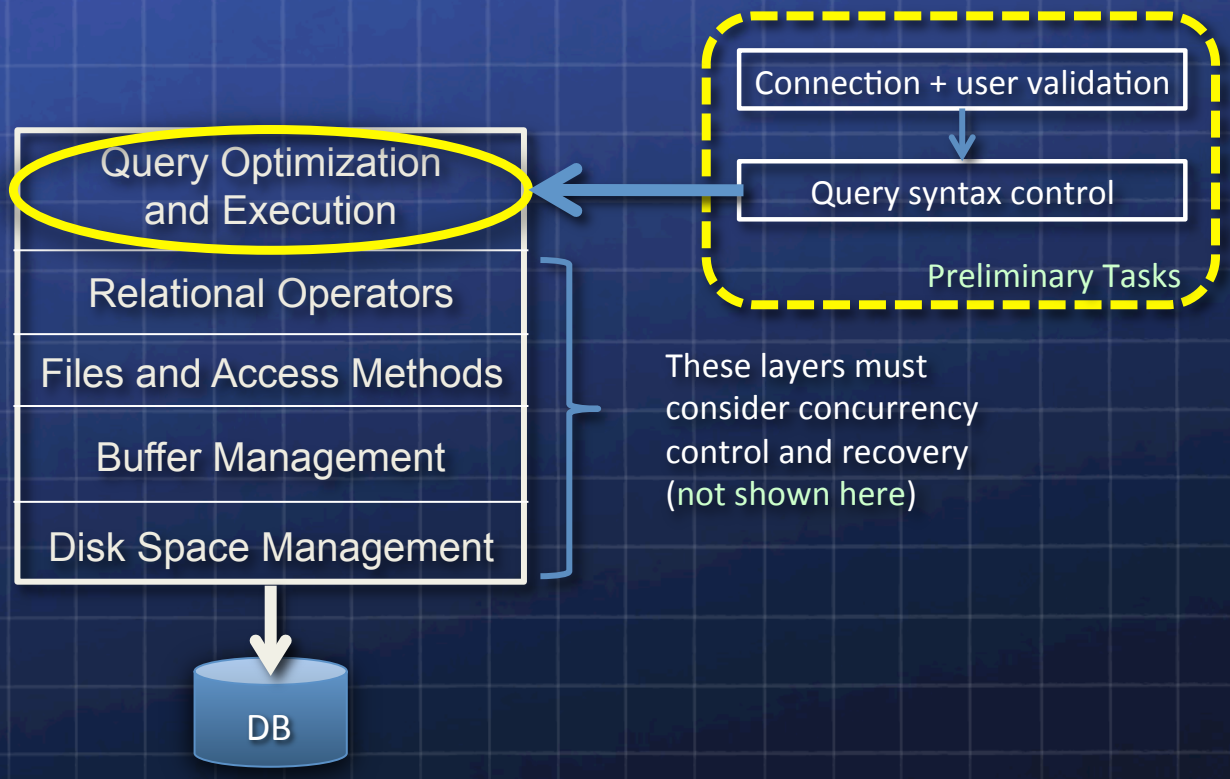
- A *data model* is a collection of concepts for describing data:
  - Structure, constraints, operations.
- A *schema* is a description of a particular collection of data, using the given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.

DBMS: who makes it happy

- End users and DBMS vendors.
- DB application programmers.
  - e.g., smart webmasters.
- *Database administrator (DBA).*
  - Designs logical / physical schemas.
  - Handles security and authorization.
  - Data availability, crash recovery.
  - Database tuning as needs evolve.

Structure of a DBMS

A typical DBMS has a layered architecture.



DBMS: summary

- DBMS used to maintain & query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.

The Messier catalogue (again)

Metadata

Id	Type	Const	Mag	Ra_h	Ra_m	Dec_d	Dec_p	Dist	App_size
int	string	string	float	float	float	float	float	string	string
1	BN	Tau	8.2	05	34.5	+22	01.0	6.3 kly	6'x4'
2	GC	Aqu	6.3	21	33.5	00	-49.0	36.2 kly	12.9'
3	GC	CVn	6.3	13	42.2	+28	23.0	30.6 kly	16.2'
4	GC	Sco	6.4	16	23.5	-26	31.5	6.8 kly	26.3'
5	GC	Ser	6.2	15	18.6	+02	05.0	22.8 kly	17.4'
6	OC	Sco	4.2	17	40.4	-32	13.8	2 kly	33'
7	OC	Sco	4.1	17	53.9	-34	47.0	800 ly	80.0'
8	BN	Sag	6.0	18	04.1	-24	18.0	5200 ly	90'x40'
9	GC	Oph	7.3	17	19.2	-18	31.0	26.4 kly	9.3'

- Let's insert it into MySQL and use it

(see ross.iasfbo.inaf.it/MCS/)

The Messier table creation

- Here we use the MySQL client command to connect to the MySQL server. Use user "root" for convenience.

```
  mysql --local-infile=1 -u root -p [ DIF ]
Enter password:
mysql> USE DIF;

mysql> SHOW TABLES;
+---------------+
| Tables_in_dif |
+---------------+
| Messier       |
| TypeDescr     |
| dif           |
| tbl           |
+---------------+
5 rows in set (0.00 sec)
```

The Messier table creation

- Drop table if already there, then create it (*no prompt*).

```
DROP TABLE IF EXISTS Messier;
CREATE TABLE Messier (
    M int NOT NULL,
    Type CHAR(2) DEFAULT '**',
    Const CHAR(3) DEFAULT '***',
    Mag FLOAT,
    Ra  FLOAT, Decl FLOAT,
    Dist CHAR(20),
    App_size CHAR(20) DEFAULT 'unknown',
        PRIMARY KEY(M),
        INDEX(Type)
);
```

The Messier table creation

- Read from an ASCII file using LOAD DATA INFILE (*no prompt*).

```
LOAD DATA LOCAL INFILE './messier'
INTO TABLE Messier;
[ SELECT * FROM Messier; ]
UPDATE Messier SET Ra = Ra * 15.0;
[ SELECT * FROM Messier; ]
```

- Also create an object type description table.

```
CREATE TABLE TypeDescr(
      Type CHAR(2), Descr CHAR(20),
      PRIMARY KEY(Type)
);
LOAD DATA LOCAL INFILE './legend'
INTO TABLE TypeDescr;
```

The Messier table creation

- The "legend" ASCII file loaded into `TypeDescr`:

```
OC  \t  Open Cluster      \n
GC      Globular Cluster
GX      Galaxy
PN      Planetary Nebula
BN      Bright Nebula
DN      Dark Nebula
*2      Double Star
*C      Star Cloud
```

Note that the default field separator is the (*invisible*) <`Tab`> (`\t`) and the end line character is <NewLine> (`\n`).

Can use any separator/terminator using the FIELDS/LINES TERMINATED BY clause. More can be given.

SQL: Structured Query Language

- Can also be used to build scripts, functions, procedures.
- Most common query statements: SELECT, INSERT, DELETE, UPDATE.
  - Note: INSERT, DELETE are "record-based" operations.
- Implemented for the first time in 1970 (with the name SEQUEL, (*Structured English Query Language*) for the IBM "System R".
- Adopted as standard ANSI (1986) and ISO (1987) with the name of SQL (pronounced "ES-Q-EL");
- There exist various versions:
  - SQL-86, SQL-87 (adopted by ISO)
  - SQL-89
  - SQL-92 (the most commonly used)
  - SQL-99 (addition of regular expressions, subqueries, triggers, etc.)
  - SQL-2003 (XML and other advanced functionalities. *Not free*!

SQL extensions and DB servers

- **Each DB server implements it's own extensions. Some are generally available, other are very specific. Relational DB servers commonly used include: Oracle, Sybase, SQLServer, DB2, MySQL, PostgreSQL, SQLite, (PicoSQL), …**
  - **We have used (and will use) MySQL as a reference. It is open source and runs on Linux, MacOS, Windows, Solaris, HPUX.**
  - ***Here we will not show the installation procedure but I suggest you try yourself and in case of troubles can contact me. Just remember to create the "root" account. It is used for all the management aspects related to the DB server.***
- **SQLite is a zero-configuration DB → no client/server architecture.**
  - **Can be used to manage Application Files.** Rather than using fopen() to write XML or some proprietary format into disk files used by your application, use an SQLite database instead. You'll avoid having to write and troubleshoot a parser, your data will be more easily accessible and cross-platform, and your updates will be transactional.

SQL: the language Statements

- **DDL**: is abbreviation of **Data Definition Language**.
  - It is used to create and modify the DB structure or schema. Examples: CREATE, ALTER, DROP.
- **DML**: abbreviation of **Data Manipulation Language**.
  - It is used for managing data within schema objects, e.g. retrieve, store, modify, delete, insert and update data. Examples: SELECT, UPDATE, INSERT.
- **DCL**: is abbreviation of **Data Control Language**.
  - It is used to create roles, permissions, and referential integrity as well as to control access to database by securing it. Examples: GRANT, REVOKE.
- **TCL**: is abbreviation of **Transactional Control Language**.
  - It is used to manage different transactions occurring within a database. Examples: COMMIT, ROLLBACK.

DDL: Data Definition Language

- Create and modify the structure of database objects in database.
  - **CREATE** - to create objects in the database
  - **ALTER** - alters the structure of the database
  - **DROP** - delete objects from the database
  - **TRUNCATE** - remove all records from a table, including all spaces allocated for the records are removed
  - **COMMENT** - add comments to the data dictionary
  - **RENAME** - rename an object

DML: Data Manipulation Language

- Manage data within schema objects.
  - **SELECT** - retrieve data from the a database
  - **INSERT** - insert data into a table
  - **UPDATE** - updates existing data within a table
  - **DELETE** - deletes all records from a table, the space for the records remain
  - **CALL** - call a SQL or (other language) subprogram
  - **LOCK TABLE** - control concurrency
  - **REPLACE/MERGE, DO, HANDLER, LOAD DATA INFILE**
  - ...

DML: Data Manipulation Language (2)

- Prepared Statements:
  - **PREPARE, EXECUTE, DEALLOCATE PREPARE**
- Compound-statements:
  - **BEGIN … END, DECLARE, SELECT … INTO**
- Cursors:
  - **OPEN, FETCH, CLOSE**
- Flow Control Constructs:
  - **IF, CASE, LOOP, LEAVE, ITERATE, REPEAT, WHILE**
  - **RETURN**

DCL: Administration Statements

- Account management: users, permissions, view status, etc.
  - **CREATE USER, DROP USER, RENAME USER, GRANT, REVOKE, SET PASSWORD, e.g.**
  - **GRANT** - gives user's access privileges to database
  - **REVOKE** - withdraw access privileges given with the GRANT command
- Table maintenance:
  - **ANALYSE TABLE, CHECK TABLE, REPAIR TABLE**
- Plugin and User-Defined Function Statements:
  - **CREATE/DROP FUNCTION, INSTALL/UNINSTALL PLUGIN**

SQL: Utility Statements and common clauses

- View status, etc.:
  - **SET, SHOW, DESCRIBE (*SHOW COLUMNS FROM*), EXPLAIN, HELP, USE, FLUSH, KILL, RESET, e.g.**
  - **SHOW *xyz*** – provide information about a long list of things: databases, tables, view, columns, functions, status information about the server, …
  - **EXPLAIN *xyz*** - explain query execution, access path to data, …
    `EXPLAIN SELECT the_Query;`
- Common SELECT clauses:
  - **LIMIT, HAVING, GROUP BY, ORDER BY, INTO OUTFILE**

Aggregation

SQL: Functions and Operators

Most of them are SQL standard, other are specific to each DB server. Below those for MySQL.

- **Operators (*with Type Conversion in Expression Evaluation*)**
- **Control Flow Functions (IF(), CASE)**
- **String Functions, Numeric Functions and Operators**
- **Date and Time Functions**
- **Full-Text Search Functions**
- **Cast Functions and Operators**
- **XML Functions**
- **Bit Functions**

…continue…

SQL: Functions and Operators

...continued

- Encryption and Compression Functions
- Information / miscellaneous Functions
- Functions and Modifiers for Use with GROUP BY Clauses
- Spatial Extensions
- Precision Math

The importance of indexing

- Except for small tables and for table which will need to be ready (almost) at every query, you must always index the table on one or more columns, either as a compound single index or with multiple indices.
- There are various types of index (or key):
  - Primary key
  - Unique key (*superkey, compound key, composite key, alternate key*)
  - Multiple key *(foreign key*)

Note:

1. Generally a key can be the combination of more columns.
2. Keys have several constraints.
3. The foreign key identifies 1 or more columns in one (referencing) table that refers to a set of columns in another (referenced) table. The columns in the referencing table must be the primary key or other candidate key in the referenced table.

The importance of indexing

Primary key constraints:

- A set of attributes is a *key* for a relation if:
  - No two distinct rows can have the same values in all key fields.
  - A proper subset of the key attributes is not a key.
- Superkey:
  - It is a set of attributes within a table whose values can be used to uniquely identify a tuple.
  - A proper subset of a superkey may be a superkey.
- If multiple keys, one of them is chosen to be the primary key.
  - e.g.: `PRIMARY KEY (Flight) = <flightNo>`

The importance of indexing

Candidate keys (SQL: Unique):

- Keys that are not primary keys are candidate keys.
- Specified in SQL using UNIQUE.
- Attribute of unique key may have NULL values !
- Eg: FlewIn (flightNo, ffNumber)

```
PRIMARY KEY (FlewIn) = <flightNo>

CANDIDATE KEY (FlewIn) = <ffNumber>
```

If a query does not use an indexed column, then the entire table will be scanned ➔ no help!

The importance of indexing

Indexing examples:

```
ALTER TABLE Messier ADD UNIQUE KEY (M);

ALTER TABLE Messier ADD INDEX (Type);

ALTER TABLE Messier ADD INDEX pos(htmID_6,Ra);
```

To show the indices:

```
SHOW INDEX FROM Messier;
```

The field to look at is Cardinality: it tells how many distinct rows that index points to in the table. The cardinality of PRIMARY / UNIQUE keys is exactly the number of rows in the table, i.e. each key points to a single entry. Each multiple key refers to Nrows / Cardinality, i.e. it gives the table grouping factor.

Example queries using indices

Examples:

```
SELECT Ra, Decl, Mag FROM Messier WHERE M=33;

SELECT M, App_size FROM Messier WHERE M BETWEEN 10 AND 13;
SELECT COUNT(*) FROM Messier WHERE Type = 'GC';

SELECT COUNT(Mag) AS N, AVG(Mag) AS Mag_ave
  FROM Messier WHERE Type ='PN';

SELECT COUNT(*) AS N, ROUND(AVG(Mag)) AS IMag_ave
  FROM Messier WHERE Type ='GX' AND Decl > 0;
```

See DIF package on:
ross.iasfbo.inaf.it/MCS

SQL functions / procedures

```
DELIMITER //
DROP FUNCTION IF EXISTS sphedist//
CREATE FUNCTION sphedist(a1 DOUBLE, b1 DOUBLE, a2 DOUBLE,
                         b2 DOUBLE) RETURNS DOUBLE
BEGIN
  DECLARE d2r, a1r, b1r, a2r, b2r, radiff, cosdis, dist DOUBLE;
  SET d2r = 0.017453292519943;
  SET a1r = a1*d2r;  SET b1r = b1*d2r;
  SET a2r = a2*d2r;  SET b2r = b2*d2r;
  SET radiff = ABS(a2r-a1r);
  IF (radiff > pi()) THEN
    SET radiff = 2e0 * pi() - radiff;
  END IF;
  SET cosdis = sin(b1r)*sin(b2r) + cos(b1r)*cos(b2r)*cos(radiff);
  SET dist = acos(cosdis)/d2r;
  RETURN dist;
END//
```

SQL functions / procedures

```
DROP PROCEDURE IF EXISTS hadec2altaz;
DELIMITER //
CREATE PROCEDURE hadec2altaz(IN ha DOUBLE, IN decl DOUBLE, IN lat
DOUBLE, OUT alt DOUBLE, OUT az DOUBLE)
BEGIN
  DECLARE d2r, sh, ch, sd, cd, sl, cl DOUBLE;
  DECLARE x, y, z, r DOUBLE;
  SET d2r = pi()/180.0;
  SET sh = sin(ha*d2r);    SET ch = cos(ha*d2r);
  SET sd = sin(decl*d2r);  SET cd = cos(decl*d2r);
  SET sl = sin(lat*d2r);   SET cl = cos(lat*d2r);
  SET x = - ch * cd * sl + sd * cl;  SET y = - sh * cd;
  SET z =   ch * cd * cl + sd * sl;  SET r = sqrt(x*x + y*y);
  SET az = atan(y,x) / d2r;  SET alt = atan(z,r) / d2r;
  IF (az < 0) THEN
    SET az = az + 360;
  END IF;
END//
```

SQL functions / procedures

Examples:

```
SELECT sphedist(170.2, 32.4, 171, 33) AS dist_arcmin;
+-------------+
| dist_arcmin |
+-------------+
|        54.1 |
+-------------+

SELECT sphedist(Ra, Decl, 83, 22)/60 AS dist_deg
  FROM Messier WHERE M=1;
+-------------------+
| dist_deg          |
+-------------------+
| 0.579695994000773 |
+-------------------+
```

SQL functions / procedures

Examples:

```
SET @ra1=0, @de1=0;
SELECT Ra, Decl INTO @ra1, @de1 FROM Messier WHERE M=1;
SELECT sphedist(Ra, Decl, @ra1, @de1)/60 AS dist_deg
  FROM Messier WHERE M>1 AND M<5;
+-----------------+
| dist_deg        |
+-----------------+
| 118.186206837217 |
| 104.661086889574 |
| 163.403397387795 |
+-----------------+
SELECT M, Type, sphedist(Ra, Decl, @ra1, @de1)/60 AS dist
  FROM Messier HAVING dist < 10;
+----+------+-----------------+
| M  | Type | dist            |
+----+------+-----------------+
|  1 | BN   |               0 |
| 35 | OC   | 8.26280575133499 |
+----+------+-----------------+
```

SQL functions / procedures

Examples:

```
CALL hadec2altaz(270, 46, 44, @alt, @az);
SELECT @alt, @az;
+------------------+------------------+
| @alt             | @az              |
+------------------+------------------+
| 29.9798507651043 | 53.3176415064057 |
+------------------+------------------+

CALL hadec2altaz(180, 46, 44, @alt, @az);
+-----------------------+------+
| @alt                  | @az  |
+-----------------------+------+
| -6.36110936292703e-15 |  360 |
+-----------------------+------+
```

Simple joins

Examples:

```
SELECT Messier.*, TypeDescr.Descr
  FROM Messier LEFT JOIN TypeDescr
  ON Messier.Type = TypeDescr.Type;
…
SELECT Messier.M, Messier.Type, Messier.App_size,
Messier.Const, TypeDescr.Descr
  FROM Messier LEFT JOIN TypeDescr
  ON Messier.Type = TypeDescr.Type LIMIT 5;
+---+------+----------+-------+------------------+
| M | Type | App_size | Const | Descr            |
+---+------+----------+-------+------------------+
| 1 | BN   | 6'x4'    | Tau   | Bright Nebula    |
| 2 | GC   | 12.9'    | Aqu   | Globular Cluster |
| 3 | GC   | 16.2'    | CVn   | Globular Cluster |
| 4 | GC   | 26.3'    | Sco   | Globular Cluster |
| 5 | GC   | 17.4'    | Ser   | Globular Cluster |
+---+------+----------+-------+------------------+
```

Aggregating

Examples:

```
SELECT propID, object, pi_coi, filter, exptime,
  COUNT(exptime) as Nobs, SUM(exptime) as TotExp,
  DATE(date_obs) as Day
FROM Obslog WHERE date_obs>'2011-04-16 22:00:00' AND
                  date_obs<'2011-04-17 12:00:00'
GROUP BY propid, object, pi_coi, filter, exptime
```

propID	object	pi_coi	filter	exptime	Nobs	TotExp	Day
5	SA098_978	REM_Team	GRI	5	7	35	2011-04-16
5	SA098_978	REM_Team	H	5	7	35	2011-04-16
5	SA098_978	REM_Team	I	20	8	160	2011-04-16
5	SA098_978	REM_Team	J	5	7	35	2011-04-16
23007	sdss15724	Elena_Pancino	R	180	3	540	2011-04-17
23007	sdss15724	Elena_Pancino	J	20	63	1260	2011-04-17
23038	DP_Cru	Marcio_Catelan	K	10	7	70	2011-04-17
23038	DP_Cru	Marcio_Catelan	V	5	1	5	2011-04-17

…

Aggregating

Examples:

```
SELECT propID, COUNT(exptime) AS Nobs, SUM(exptime) AS
TotExp
FROM Obslog WHERE date_obs>'2011-04-16 22:00:00' AND
                  date_obs<'2011-04-17 12:00:00'
GROUP BY propID ORDER BY TotExp [ ASC | DESC ];
```

propID	Nobs	TotExp
23071	24	24
7	11	55
5	59	655
23012	30	960
23062	22	1470
23073	24	1530
23030	87	1722
23735	120	2120
23038	120	2475
23007	72	2880
23027	117	3510
23011	196	5880
23047	210	6300

Views

```
DROP VIEW IF EXISTS Messier_view;
CREATE VIEW Messier_view AS
  SELECT Messier.M,
         TypeDescr.Descr AS Descr,
         Messier.Const,
         Messier.Mag,
         Messier.Ra,
         Messier.Decl,
         radec2alt(Messier.Ra, Messier.Decl, 44.5, 11.3, -1,
                   now()) AS Alt,
         radec2az( Messier.Ra, Messier.Decl, 44.5, 11.3, -1,
                   now()) AS Az,
         Messier.Dist,
         Messier.App_size
  FROM Messier LEFT JOIN TypeDescr
  ON Messier.Type = TypeDescr.Type
  ORDER BY Descr, M;
```

Views

```
DESCRIBE Messier_view;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| M        | int(11)  | NO   |     | NULL    |       |
| Descr    | char(20) | YES  |     | NULL    |       |
| Const    | char(3)  | YES  |     | ***     |       |
| Mag      | float    | YES  |     | NULL    |       |
| Ra       | float    | YES  |     | NULL    |       |
| Decl     | float    | YES  |     | NULL    |       |
| Alt      | double   | YES  |     | NULL    |       |
| Az       | double   | YES  |     | NULL    |       |
| Dist     | char(20) | YES  |     | NULL    |       |
| App_size | char(20) | YES  |     | unknown |       |
+----------+----------+------+-----+---------+-------+
```

Views

```
SELECT * FROM Messier_view limit 4;

| M  | Descr          | Const | Mag  | Ra       | Decl      | Alt
                  | Az                | Dist      | App_size |

| 73 | NULL           | Aqu   |    9 | 314.726 |  -11.365 |
-62.218887202434  | 341.173761681747 | --        | 2.8'     |

|  1 | Bright Nebula | Tau   |  8.2 |  83.625 |  22.0167 |
50.4744339376439  | 258.497278567727 | 6.3 kly   | 6'x4'    |

|  8 | Bright Nebula | Sag   |    6 | 271.025 |    -23.7 |
-57.078773558122  | 74.4445883563204 | 5200 ly   | 90'x40'  |

| 17 | Bright Nebula | Sag   |  7.5 | 275.175 | -15.8333 |
-55.0517205008551 | 59.1020440792511 | 5000 ly | 11       |
```

How to insert data files into a DB

- For ASCII (text) files several methods are available within the various DB server. One seen. A lot of external tools also exist.
- For FITS files see MCS.
- It is easy to write simple programs to import into DBs binary files. Can make an exercise.

How to export data from one DB to another

- There are many possibilities, depending if a full table has to be exported or just a subset:
  - Direct transfer: INSERT INTO DB2.tab1 SELECT * FROM DB1.tab1
- If different server are involved then it depends on the size of the data to export. It is possible to:
  - On server A export to a file the data and then restore them on the server B.
  - From server B perform the query on server A and store/save the data (all at once or in chunks) into a local file or directly into the DB table.
  - Simply copy the DB disk files from server A into the DB disk on server B (a DB = directory, a table = one or more files).
  - Write a program that connects to the two DBs (read from A write to B).

Not the end…

On request, one more seminar on specific a issue can be agreed.

Email: nicastro@iasfbo.inaf.it