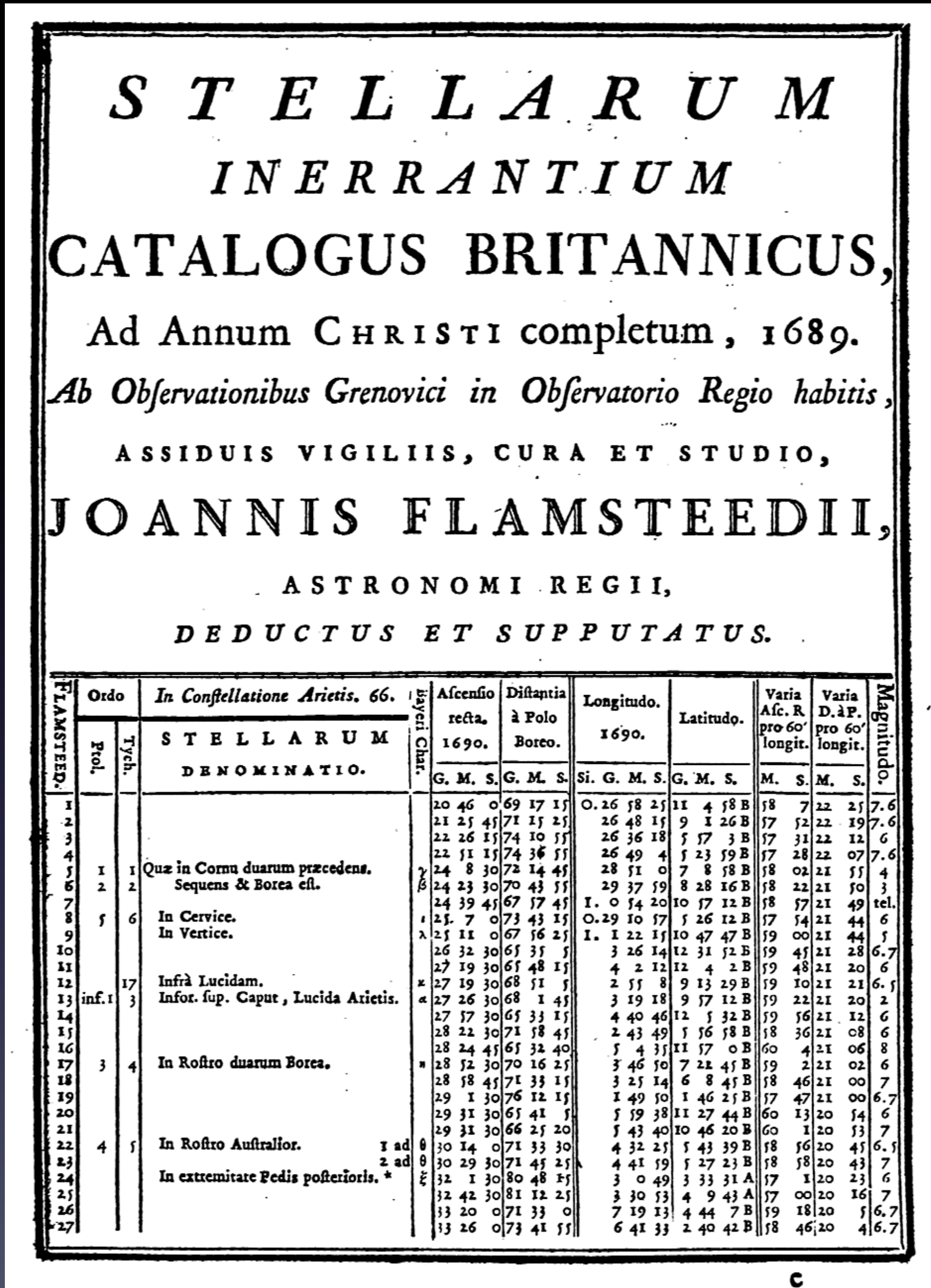
Databases

Why Use (Relational) Databases in Astronomy?

- Much of astronomy, and especially survey astronomy, begins with *collecting sets of well defined measurements* on samples (or entire populations) of objects
- We organize and publish these measurements as astronomical catalogs
  - These are collections of tables
  - Used to be published as (big, thick!) books



STELLARUM INERRANTIUM CATALOGUS BRITANNICUS,

Ad Annum CHRISTI completum, 1689.

Ab Observationibus Grenovici in Observatorio Regio habitis,

ASSIDUIS VIGILIIS, CURA ET STUDIO,

JOANNIS FLAMSTEEDII,

ASTRONOMI REGII,

DEDUCTUS ET SUPPUTATUS.

FLAMSTEED.	Ordo		*In Constellatione Arietis. 66.*	Bayeri Char.	Ascensio recta. 1690.			Distantia à Polo Boreo.			Longitudo. 1690.				Latitudo.			Varia Asc. R pro 60′ longit.		Varia D. à P. pro 60′ longit.		Magnitudo.
	Prol.	Tych.	STELLARUM DENOMINATIO.		G.	M.	S.	G.	M.	S.	Si.	G.	M.	S.	G.	M.	S.	M.	S.	M.	S.	
1					20	46	0	69	17	15	0.	26	58	25	11	4	58 B	58	7	22	25	7.6
2					21	25	45	71	15	25		26	48	15	9	1	26 B	57	52	22	19	7.6
3					22	26	15	74	10	55		26	36	18	5	57	3 B	57	31	22	12	6
4					22	51	15	74	36	55		26	49	4	5	23	59 B	57	28	22	07	7.6
5	1	1	Quæ in Cornu duarum præcedens.	γ	24	8	30	72	14	45		28	51	0	7	8	58 B	58	02	21	55	4
6	2	2	Sequens & Borea est.	β	24	23	30	70	43	55		29	37	59	8	28	16 B	58	22	21	50	3
7					24	39	45	67	57	45	I.	0	54	20	10	57	12 B	58	57	21	49	tel.
8	5	6	In Cervice.	ι	25.	7	0	73	43	15	0.	29	10	57	5	26	12 B	57	54	21	44	6
9			In Vertice.	λ	25	11	0	67	56	25	I.	1	22	15	10	47	47 B	59	00	21	44	5
10					26	32	30	65	35	5		3	26	14	12	31	52 B	59	45	21	28	6.7
11					27	19	30	65	48	15		4	2	12	12	4	2 B	59	48	21	20	6
12		17	Infrà Lucidam.	κ	27	19	30	68	51	5		2	55	8	9	13	29 B	59	10	21	21	6.5
13	inf. 1	3	Infor. sup. Caput, Lucida Arietis.	α	27	26	30	68	1	45		3	19	18	9	57	12 B	59	22	21	20	2
14					27	57	30	65	33	15		4	40	46	12	5	32 B	59	56	21	12	6
15					28	22	30	71	58	45		2	43	49	5	56	58 B	58	36	21	08	6
16					28	24	45	65	32	40		5	4	35	11	57	0 B	60	4	21	06	8
17	3	4	In Rostro duarum Borea.	η	28	52	30	70	16	25		3	46	50	7	22	45 B	59	2	21	02	6
18					28	58	45	71	33	15		3	25	14	6	8	45 B	58	46	21	00	7
19					29	1	30	76	12	15		1	49	50	1	46	25 B	57	47	21	00	6.7
20					29	31	30	65	41	5		5	59	38	11	27	44 B	60	13	20	54	6
21					29	31	30	66	25	20		5	43	40	10	46	20 B	60	1	20	53	7
22	4	5	In Rostro Australior. 1 ad	θ	30	14	0	71	33	30		4	32	25	5	43	39 B	58	56	20	45	6.5
23			2 ad	θ	30	29	30	71	45	25		4	41	59	5	27	23 B	58	58	20	43	7
24			In extremitate Pedis posterioris. *	ξ	32	1	30	80	48	15		3	0	49	3	33	31 A	57	1	20	23	6
25					32	42	30	81	12	25		3	30	53	4	9	43 A	57	00	20	16	7
26					33	20	0	71	33	0		7	19	13	4	44	7 B	59	18	20	5	6.7
27					33	26	0	73	41	55		6	41	33	2	40	42 B	58	46	20	4	6.7

c

Left: The first page of J. J. Lalande's edited and corrected version of John Flamsteed's star catalogue, published in 1783. The stars shown here belong to the constellation Aries. In the first column, Lalande numbered each star consecutively by constellation. These are the numbers that we now call Flamsteed numbers.

From http://www.ianridpath.com/startales/flamsteed.htm

```
I    photo.in
7757,301,1,74,186,6,8.12944435106658,26.6266172894736,17.04889,18.16535,0.01654805,0.02145229
7757,301,1,74,187,6,8.12783867556709,26.627245975921,17.37402,17.92875,0.02894481,0.02568013
7757,301,1,74,188,3,8.1273232524192,26.6251199416623,20.1466,21.35297,0.3003744,0.3302762
4288,301,1,39,682,3,24.5161170422305,-1.16579446393527,22.97032,24.3259,0.2672399,0.5240437
4288,301,1,39,683,3,24.5179406515354,-1.1792069022485,22.62052,25.09109,0.1850479,0.6585805
4288,301,1,39,684,6,24.5189463293148,-1.15915086108891,21.4247,23.04125,0.06608655,0.1968172
4136,301,1,61,935,6,36.4715922759092,-1.06093938828308,22.71782,23.14112,0.158014,0.1799687
4136,301,1,61,936,3,36.4717583013136,-1.1378448207726,22.81683,23.88123,0.1742272,0.3260605
4136,301,1,61,937,3,36.4717582434391,-1.13784497192974,22.81147,23.87586,0.1734457,0.3247895
4288,301,1,40,311,3,24.6839203338022,-1.23631696217547,21.2002,21.67521,0.05694564,0.06316777
4288,301,1,40,312,3,24.6840602692246,-1.21784918362007,20.30287,21.04976,0.02972161,0.04032911
4288,301,1,40,313,6,24.6840216690377,-1.08292772289886,24.92263,25.72778,0.68427,0.5471938
5598,301,1,61,792,3,351.787950113407,6.14573538435867,22.43574,23.83793,0.1125768,0.2867745
5598,301,1,61,793,3,351.787950113434,6.14573538316393,22.43573,23.77753,0.1174541,0.2741905
5598,301,1,61,794,6,351.787349107439,6.14481612145222,24.6701,24.8507,0.4675894,0.4849904
2699,301,1,48,527,3,12.0760019016408,-3.32677418219699,22.18116,23.27577,0.126546,0.2270369
2699,301,1,48,528,6,12.0770027529666,-3.32913243320258,22.12757,23.79366,0.1215217,0.3403472
2699,301,1,48,529,3,12.0832728187538,-3.52539818226738,22.29741,23.32008,0.1377919,0.2253349
94,301,1,38,279,6,340.524768659138,-0.843090883870374,20.75028,21.13888,0.04839022,0.04644739
94,301,1,38,280,6,340.525793656628,-0.965210498356983,24.23321,26.07633,0.8058653,0.7702702
94,301,1,38,281,6,340.53257887691,-1.02365035629542,21.10608,21.15248,0.06605724,0.04672106
4288,301,1,76,766,3,30.0899167300738,-1.25189466355601,22.57054,22.91144,0.1807321,0.1984752
4288,301,1,76,767,3,30.0899962733195,-1.14314301954175,21.99419,23.78533,0.1094794,0.4162939
4288,301,1,76,768,6,30.0899156247035,-1.19236549812758,22.64196,22.91776,0.1927483,0.2022232
7937,301,1,84,354,3,5.5913226470183,26.6120638856974,22.63659,23.76799,0.2950225,0.5986399
7937,301,1,84,355,3,5.58984744314193,26.6157353187021,21.27009,22.13992,0.08808753,0.1490507
7937,301,1,84,356,6,5.59007089167375,26.5703384153732,24.52393,26.33231,1.060107,0.6528299
3996,301,1,81,615,6,216.223187662076,11.9472286471793,19.93735,21.0716,0.02294876,0.03470354
3996,301,1,81,616,3,216.23835672109,12.0534224512804,20.22574,21.24497,0.02592515,0.03942162
3996,301,1,81,617,3,216.219379232008,11.9171402588482,22.60762,23.05279,0.1416738,0.1555459
6354,301,1,29,2997,6,332.302798261878,41.2179102291468,20.95895,22.51458,0.04065189,0.1163793
6354,301,1,29,2998,6,332.355200585822,41.2447081319807,20.78153,22.35575,0.03523929,0.09754675
6354,301,1,29,2999,3,332.38785409452,41.2613314909198,21.00566,22.75478,0.04217106,0.1353208
2986,301,1,59,251,6,127.586395164415,2.97627035997022,24.42951,26.40598,0.6053675,0.9018285
2986,301,1,59,252,6,127.585997207459,2.98207493069978,25.42876,25.06479,0.5993609,0.8693208
2986,301,1,59,253,6,127.588133654789,2.
5598,301,1,31,1254,6,347.289019727591,6
5598,301,1,31,1255,6,347.29286806028,6
5598,301,1,31,1256,6,347.294399617648,6
```

1,231,051,050 rows (SDSS DR10, PhotoObjAll table)

~500 columns

Why Databases for Astronomy, contd.

- There's the general challenge of organizing the collected data
  - Especially acute for large surveys
  - Can also be problematic even for small studies

- Publishing and sharing (subsets of) survey data
  - Large surveys publish billions of rows and hundreds of measurements for each observation. The catalogs are ~1-10s of TB range.
  - But not all science cases care about every kind of measurement

- The data in the catalogs is just a starting point for further work. That work can involve:
  - Statistically summarizing the properties of the data (counts, means, medians, etc.)
  - Selecting subsets of the objects or their measurements ("slicing and dicing")
  - Cross-correlating various properties (i.e., positional matching)
  - Transforming the data in various ways (e.g. from instrumental to calibrated magnitudes, accounting for reddening, etc.)
  - Identifying rare, interesting, or unusual objects

- Storing the data into a database is the industry standard way to solve these types of problems

What Are Databases?

- Logically
  - Organized collections of data
  - Typically, a set of tables and their relationships ("relational databases")
    - Terminology: for practical purposes, relation == table. For details, see http://en.wikipedia.org/wiki/Relation_%28database%29
  - A table is made up of rows and columns
    - Each row can be considered as an entry corresponding to some real-world object, listing its attributes
    - Columns define the attributes; each column has a well defined data type (e.g., integer, real, text, etc.)

- Physically
  - A collection of files written in special format, that are accessed and manipulated using a *Database Management System* (DBMS)

Examples

Person

Login	LastName	FirstName
skol	Kovalevskaya	Sofia
mlom	Lomonosov	Mikhail
dmitri	Mendeleev	Dmitri
ivan	Pavlov	Ivan

Project

ProjectId	ProjectName
1214	Antigravity
1709	Teleportation
1737	Time Travel

Experiment

ProjectId	ExperimentId	NumInvolved	ExperimentDate	Hours
1214	1	1	NULL	1.5
1214	2	1	1889-11-01	14.3
1709	1	3	1891-01-22	7.0
1709	2	1	1891-02-23	7.2
1737	1	1	1900-07-05	-1.0
1737	2	2	1900-07-05	-1.5

Involved

ProjectId	ExperimentId	InvolvedId	Login
1214	1	1	mlom
1214	2	1	mlom
1709	1	1	dmitri
1709	1	2	skol
1709	1	3	ivan
1709	2	1	mlom
1737	1	1	skol
1737	2	1	skol
1737	2	2	ivan

http://goo.gl/jWDIzy

Table: runs   New Record   Delete Record

	run	ra	dec	mjdstart	mjdend	node	inclination	mu0	nu0
1	94	336.4327791...	-1.04429400...	51075.23321...	51075.45501...	286.855205	0.009477	336.4326667...	-1.05150869...
2	109	396.2418087...	-1.25055686...	51078.39078...	51078.47494...	283.3917469...	0.008279	36.24187915...	-1.25818616...
3	125	350.4697426...	-1.25274979...	51081.25575...	51081.49528...	287.818732	0.007781	350.4696642...	-1.25966106...
4	211	402.5811092...	-1.26517002...	51115.307	51115.46205...	283.2197800...	0.007975	42.58119581...	-1.27212059...
5	240	375.1896778...	-1.26440348...	51132.185032	51132.24885...	290.578187	0.010103	15.18965685...	-1.27446183...
6	241	403.0295478...	-1.26513669...	51132.26214...	51132.30359...	266.7155050...	0.005148	43.02963017...	-1.26869244...
7	250	15.35717871...	-1.03608421...	51133.183	51133.36699...	62.095899	0.024055	15.35688309...	-1.01856644...
8	251	85.88000457...	-1.00945333...	51133.37808...	51133.40792...	11.252511	0.037496	85.87982628...	-1.04560781...
9	256	-8.28409345...	-1.05720709...	51134.11449...	51134.13357...	58.141704	0.024019	351.7157311...	-1.03519263...
10	259	368.3751608...	-1.04718589...	51134.16041...	51134.39053...	299.408811	0.007597	8.375110834...	-1.05427670...
11	273	371.5027215...	-1.25773504...	51136.164	51136.38085...	286.5415300...	0.008068	11.50270590...	-1.26577186...
12	287	396.4868469...	-1.15429721...	51138.22760...	51138.40424...	295.298232	0.007857	36.48687773...	-1.16200488...
13	297	61.15102149...	-1.15372111...	51139.293	51139.37260...	92.038416	0.040845	61.15032199...	-1.13275302...

< 1 - 14 of 765 >   Go to: 1

```
mjuric@gallifrey:~/projects/class3$ ls -l *.db
-rw-r--r--  1 mjuric  staff  5271552 Jan 21 06:14 class3.db
mjuric@gallifrey:~/projects/class3$
```

When do you need a Database

- Multiple simultaneous changes to data (concurrency)
- Data changes on a regular basis
- Large data sets where you only need some observations/variables
- Share huge data set among many people
- Rapid queries with no analysis
- Web interfaces to data, especially dynamic data

P. Spector @ Berkeley (http://goo.gl/YoPVyr)

Becoming ubiquitous today

- Large surveys today typically make their data available as a remotely queryable relational database
  - A typical user does not need all the data, but just a subset. Keeping it in a database allows the scientist to ask for that (custom-made) subset.
    - N.b. still not all the way there (more later!)
- Examples:
  - SDSS, WISE, PanSTARRS (soon), Gaia (soon-ish)

Personal Research: Databases vs. Files

- The “old way”:
  - store data tables as text files
  - write interpreters to read/write each format
  - write filters to select data that I want
  - write matching algorithms when I need to search multiple files (or make large franken-tables)
  - data and code are co-dependant and generally hard to keep in sync

h.t. Paul Hancock, ICRAR

Personal Research: Databases vs. Files

- The “new way”
  - one database
  - multiple tables
  - read/write translation is done by the db
  - matching tables is done by the db
  - selecting/filtering data is done by the db
  - db and code are no longer locked together

h.t. Paul Hancock, ICRAR

Interacting with Databases: Database Management Systems (DBMS)

- As mentioned before, a database can logically be thought of as a set of tables. Physically (on disk) it's stored as one or more files. They're written in a special format that cannot be directly read or written.

- A ***Database Management System (DBMS)*** is needed to read and write it
  - A software product tool that allows us to read or write data in databases
  - It allows us to query for and retrieve (a potentially transformed) subset of data from one or more tables

- Note: the on-disk format is DBMS-specific

Common DBMS

- SQLite
  - http://sqlite.org
  - Easy to use, simple, reasonably fast, free
  - Comes with Anaconda, included in Python
  - The database is a single file
  - No need for special accounts, permissions, or servers
  - GUI: http://sqlitebrowser.org
  - Downsides:
    - Poor multi-user support
    - Does not scale well (won't scale to tens or hundreds of millions of rows)

Common DBMS

- MariaDB (also, MySQL)
  - http://mariadb.org
  - Free, secure, scalable
  - Widely used and well supported
    - Comes in nearly all Linux distributions
    - There's no question that hasn't already been asked on StackOverflow ☺
  - Client/server architecture
  - More advanced features compared to SQLite
  - Can handle tables with billions of rows
  - MariaDB vs MySQL: use MariaDB
  - Will be used by LSST to serve its petabyte-scale dataset
  - Disadvantages:
    - Steeper learning curve, more initial setup

Common DBMS

- PostgreSQL
  - http://postgresql.org
  - Free, secure
  - Similar to MySQL in terms of functionality
  - Some features are more advanced, performance can be better
  - Smaller community (though still widely used), steeper learning curve

Common DBMS

- MS SQL Server
  - Not free, but performant and scalable
  - Used by the SDSS archive
- Oracle Database
  - The "industry standard" for mission critical databases
  - (Very) expensive

- ***Typically, there's no need to use a commercial solution today, except in very specialized circumstances – the free/open source databases work well enough***
- ***In this course, we will generally use sqlite (and a different solution called LSD; more later)***

Non-Traditional DBMS

- “NoSQL” databases
- Databases for storing less structured or unstructured data (e.g., web pages)
- **Very fast**, **very scalable** (>petabytes of data), do not require fixed table schemas, avoid joins
- Examples: MongoDB, Couchbase, Riak, Cassandra, Redis, CouchDB, HBase, …
- Disadvantages:
  - More difficult to work with and primitive compared to relational databases
  - Less expressive query languages, require programming for most tasks

Using a Database: SQL

- **SQL**, or **Structured Query Language** is a special-purpose programming language designed for handling data managed by relational database management systems
- It is a language that virtually all databases “speak”
  - Allows one to ask for subsets of data, join tables, modify the outputs, as well as add and delete data in the database
  - Note: there are dialects and small differences from database to database

Get sqlitebrowser

- http://sqlitebrowser.org

SQL Basics

- CREATE
  - Creating tables
- INSERT/DELETE
  - Adding and deleting rows

- SELECT
  - Selecting a subset of data
  - Joining (combining) data from different tables

- More information: http://robots.thoughtbot.com/back-to-basics-sql

Creating a (sqlite) Database

- The details of database creation and data import are DBMS specific, but the general idea is similar:
  1. Create the database itself
  2. Create the tables within the database
  3. Import the data

The "mini SDSS" Database

- Sample data:
  - See notebooks/databases/* in the class git repository
  - I extracted a random sample of ~50,000 objects from SDSS DR10 PhotoObjAll table. This is the catalog of all sources that the SDSS has detected and measured. The result is in sample.csv.
  - I also have a list of SDSS "runs" (observations) with details about each run (runs.txt)
  - I will import these two into a sqlite database

#1. Create the tables

```
CREATE TABLE sources (
        run             INTEGER,
        rerun           INTEGER,
        camcol          INTEGER,
        field           INTEGER,
        obj             INTEGER,
        type            INTEGER,
        ra              REAL,
        dec             REAL,
        psfMag_r        REAL,
        psfMag_g        REAL,
        psfMgErr_r      REAL,
        psfMagErr_g     REAL
);
```

```
CREATE TABLE runs (
        run             INTEGER,
        ra              REAL,
        dec             REAL,
        mjdstart        REAL,
        mjdend          REAL,
        node            REAL,
        inclination     REAL,
        muo             REAL,
        nuo             REAL
);
```

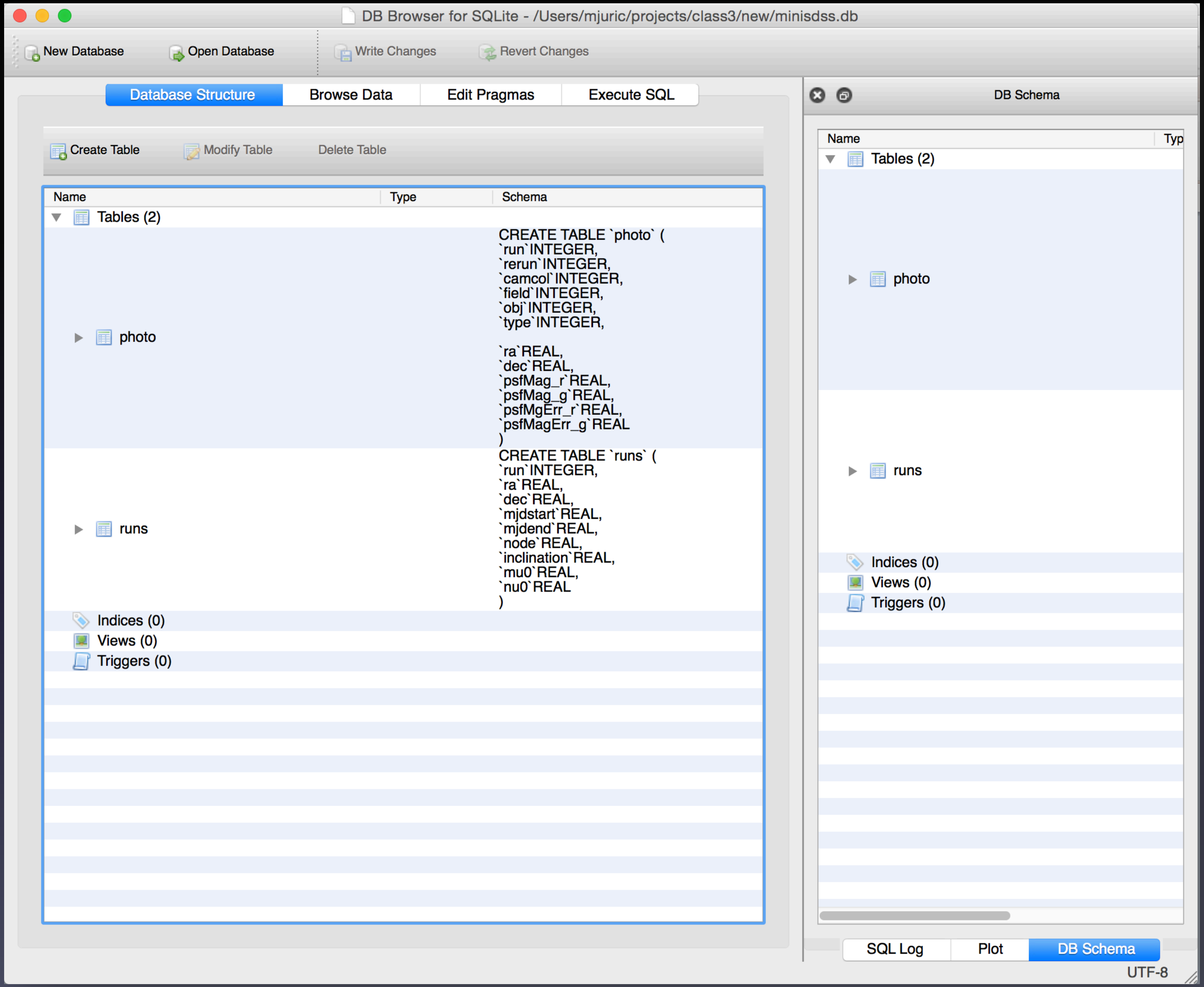
#2. Prepare the input data

- Need to do some editing to remove the headers

#2. Import

```
sqlite> .mode csv
sqlite> .separator " "
sqlite> .import runs.in runs
sqlite> .separator ","
sqlite> .import sample.in sources
sqlite> .quit
```

open -a "DB Browser for SQLite" sdss.db



SELECT Statement

- SELECT ra, dec, psfMag_r FROM sources
- SELECT ra, dec, psfMag_r FROM sources WHERE psfMag_r < 21.5
- SELECT ra, dec, psfMag_r FROM sources WHERE psfMag_r < 21.5 LIMIT 5
- SELECT COUNT(psfMag_r), AVG(psfMag_r) FROM sources WHERE psfMag_r < 21.5
- SELECT COUNT(*), run FROM sources GROUP BY run
- SELECT COUNT(*), run FROM sources GROUP BY run ORDER BY run
- SELECT COUNT(*) as ct , run FROM sources GROUP BY run ORDER BY ct

NULL

- How do we mark missing data?
  - Typical way to do this is to designate a value as “magic”
  - E.g.,: -9999 in our example database
- Relational databases provide us with a special constant, a “NULL”
  - The meaning is always clear (i.e. – no data)
  - Plays well with aggregate functions
    - I.e., AVG(), COUNT() ignore null values

UPDATE

- UPDATE sources

  SET psfMag_r = NULL

  WHERE psfMag_r = -9999.0

The table to update

Columns to update (and the values to use)

Selecting the subset of rows to update

JOIN: Joining tables

- Example:
  - Each row in the ‘sources’ table has a `run’ entry – the ID of the SDSS run where this object was observed
  - Each entry in the ‘runs’ table has a ‘mjdstart’ entry, indicating the time when the observing for this run started
  - How can we find the mjdstart for each object? An algorithm for doing it by hand:
    - For each row in the sources table:
      - Read off the value of ‘run’
      - Find the corresponding row in the ‘runs’ table
      - Read off the value of mjdstart

JOIN: Joining tables

- SELECT

  sources.ra, sources.dec, sources.run, mjdstart

  FROM

  sources

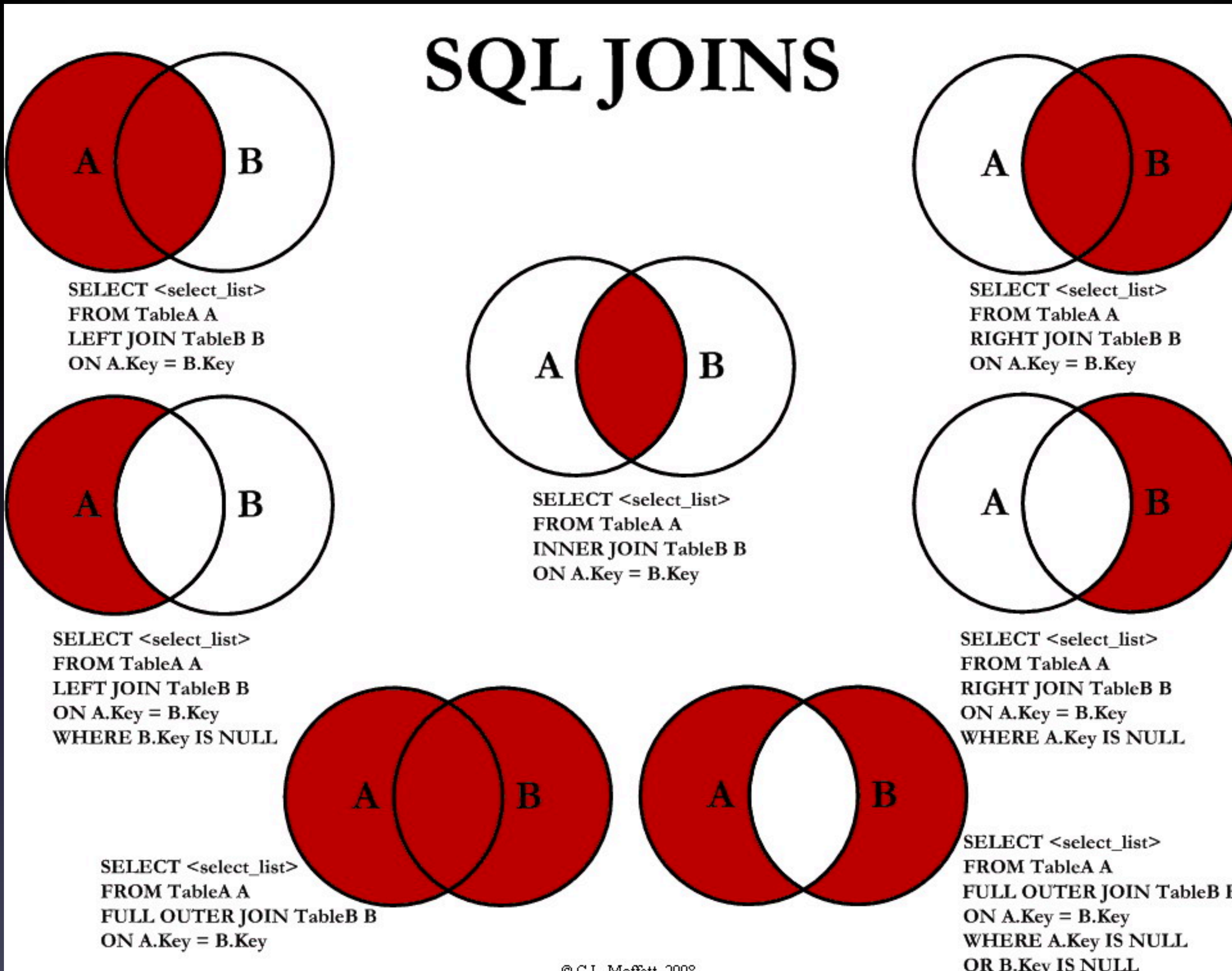
  JOIN runs ON sources.run = runs.run

The columns we're interested in.

Those appearing in more than one table need to be prefixed by the table name.

The table we're querying

Instructions how to join the runs table onto the sources table.



http://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg

Doing all of this from Python

- Python can connect to a variety of databases
- SQLite module comes built into Python (sqlite3)

- We will also use a library called pandas ("Python Data Analysis Library")
  - http://pandas.pydata.org
  - Pandas provides high-performance data structures for manipulating and analyzing tabular data

More about SQL & Databases

- Interactive SQL tutorial
  - http://sqlzoo.net/wiki/Main_Page
- Introduction to SQL (Stanford)
  - https://class.stanford.edu/courses/DB/SQL/SelfPaced/courseware/ch-sql/seq-vid-introduction_to_sql/
- Introduction to SQL (Phil Spector, Berkeley)
  - https://www.stat.berkeley.edu/~spector/sql.pdf

- Databases in depth: CSE444
  - http://courses.cs.washington.edu/courses/cse444/

Next time

- More about Pandas & hands-on with databases

Please read before Thursday

- Pandas
  - 10 minute tutorial: http://pandas.pydata.org/pandas-docs/stable/10min.html
  - 10 minute tutorial video: http://vimeo.com/59324550
  - Pandas Tutorials: http://pandas.pydata.org/pandas-docs/stable/tutorials.html

Please Do Before Thursday

- Get an ccount on SDSS Catalog Access Server
  - http://skyserver.sdss.org/casjobs/