Data Management Part 1

Yusra AlSayyad
LSST Data Management
Princeton University

DSFP- Session 3
April 25 2017

Why do astronomers care about data management?

- Bigger project teams
- Bigger Datasets

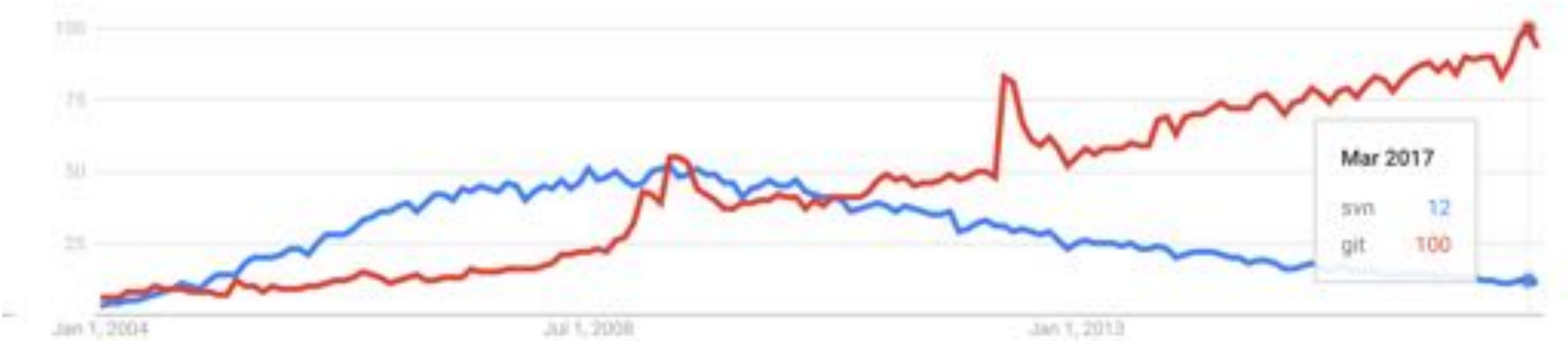
- Data is valuable but time-consuming/costly to manage

- Data Science:
  - Prepping to run the model: *"Gathering cleaning matching integrating restructuring transforming loading filtering combining merging verifying extracting" - Bill Howe*
  - Running the model
  - Communicating results

- Goal: recognize what kind of data you have, decide what tools to use

Focus on Data Management Ideas

Google Trends for svn vs git



Today’s topics

- SQL Databases in broader context
- Relational Algebra (operations on tables)
- Alternative implementations of Relational Algebra
- Indexes (and Spatial Indexes)

- TOMORROW:
  - Scaling out
  - Parallel databases (including LSST’s Qserv), NoSQL databases,
  - Summary of options available and tradeoffs to consider

Databases in Context

Databases fit into a larger context of data management

- How do we store data?

Physically?

Logically?

What would you do with a million images?

```
|--decam
|--gapon
|--hsc
|    |--calib
|    |--raw
|    |--repo
|    |    |--CFHTLS_W1
|    |    |--COMET2014F3
|    |    |--COSMOS
|    |    |--DARK
|    |    |--DEEPE01
|    |    |--DEEPE02
|    |    |--DEEPE03
|    |    |--DEEPE04
|    |    |--DEEPE05
|    |    |--DEEPE06
|    |    |--DEEPE07
|    |    |--DEEPE08
|    |    |--DEEPE09
|    |    |--DEEPE10
|    |    |--DEN_A
|    |    |--DEN_C
|    |    |--DEN_E
|    |    |--DITH_14H
```

What would you do with a million images?

```
|--decam
|--gapon
|--hsc
|   |--calib
|   |--raw
|   |--repo
|   |   |--CFHTLS_W1
|   |   |--COMET2014F3
|   |   |--COSMOS
|   |   |   |--2014-02-02
|   |   |   |--2014-02-05
|   |   |   |--2014-03-26
|   |   |   |--2014-03-27
|   |   |   |--2015-01-17
|   |   |   |--2015-01-18
|   |   |   |--2015-01-19
|   |   |   |--2015-01-20
|   |   |--DARK
|   |   |--DEEPE01
|   |   |--DEEPE02
|   |   |--DEEPE03
|   |   |--DEEPE04
|   |   |--DEEPE05
|   |   |--DEEPE06
```

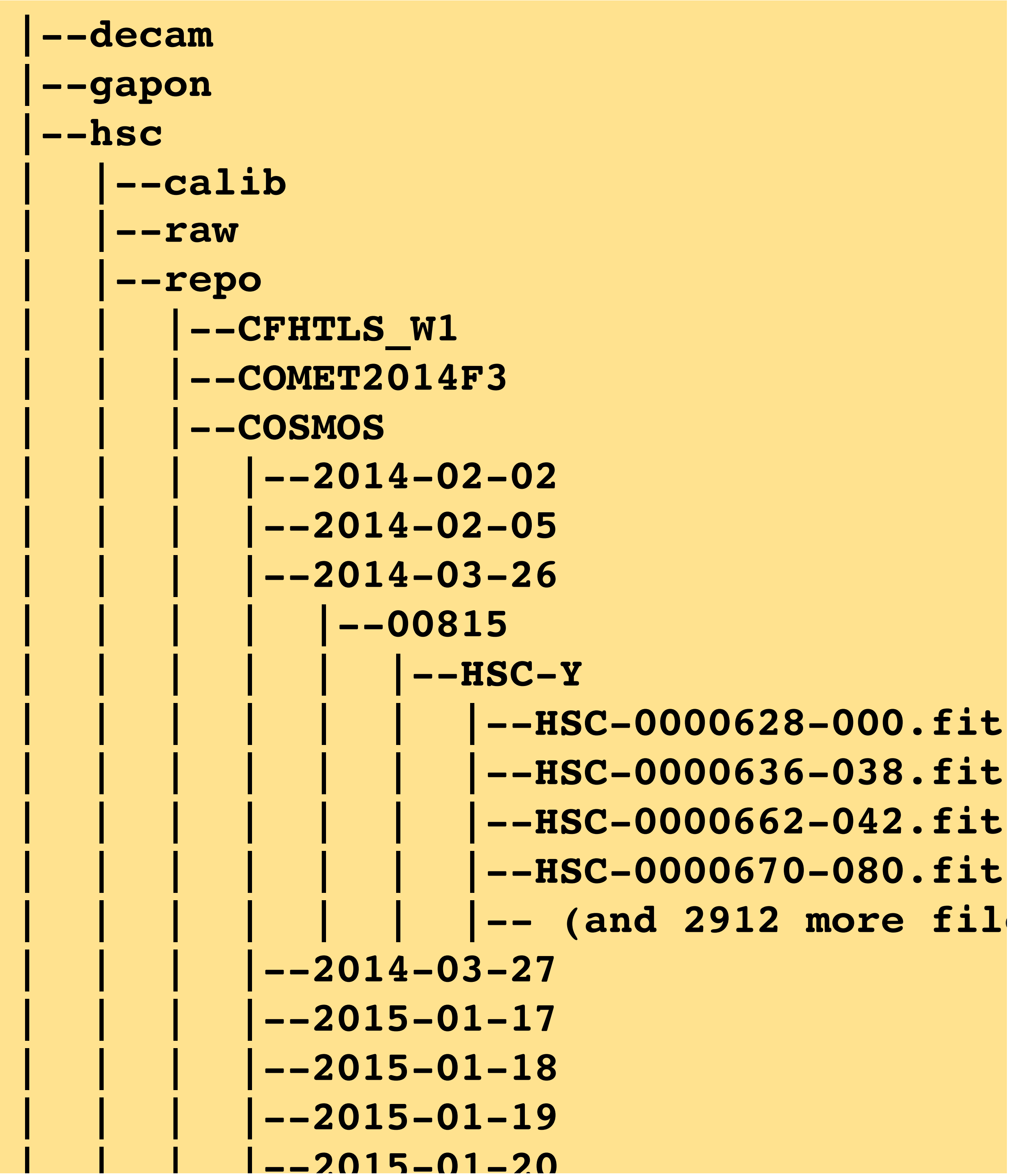
Time to count all images taken:

With HSC

in the COSMOS field,

on March 26th 2014:

```
$ time ls
hsc/repo/COSMOS/2014-
03-26/00815/HSC-Y |
wc -l

2912

real    0m0.039s
user    0m0.029s
sys    0m0.008s
```

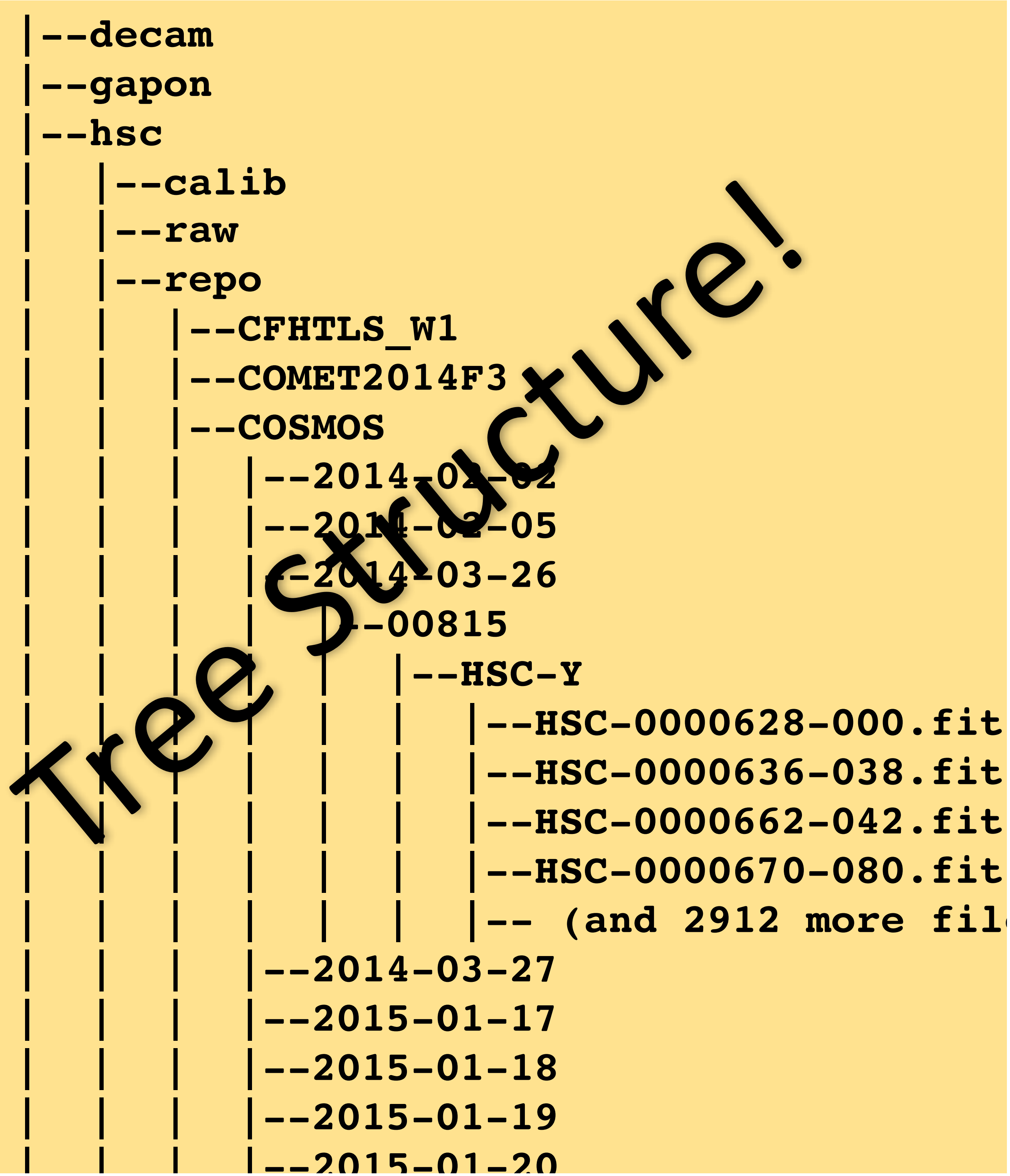


```
|--decam
|--gapon
|--hsc
|  |--calib
|  |--raw
|  |--repo
|  |  |--CFHTLS_W1
|  |  |--COMET2014F3
|  |  |--COSMOS
|  |  |  |--2014-02-02
|  |  |  |--2014-02-05
|  |  |  |--2014-03-26
|  |  |  |  |--00815
|  |  |  |  |  |--HSC-Y
|  |  |  |  |  |  |--HSC-0000628-000.fit
|  |  |  |  |  |  |--HSC-0000636-038.fit
|  |  |  |  |  |  |--HSC-0000662-042.fit
|  |  |  |  |  |  |--HSC-0000670-080.fit
|  |  |  |  |  |  |-- (and 2912 more fil
|  |  |  |--2014-03-27
|  |  |  |--2015-01-17
|  |  |  |--2015-01-18
|  |  |  |--2015-01-19
|  |  |  |--2015-01-20
```

Data Models are Unavoidable

All images taken with the HSC-Y band?

```
$ time ls
hsc/repo/*/*/*/HSC-
Y | wc -l
193171

Real   48m8.737s
user   0m5.824s
sys    1m2.072s
```

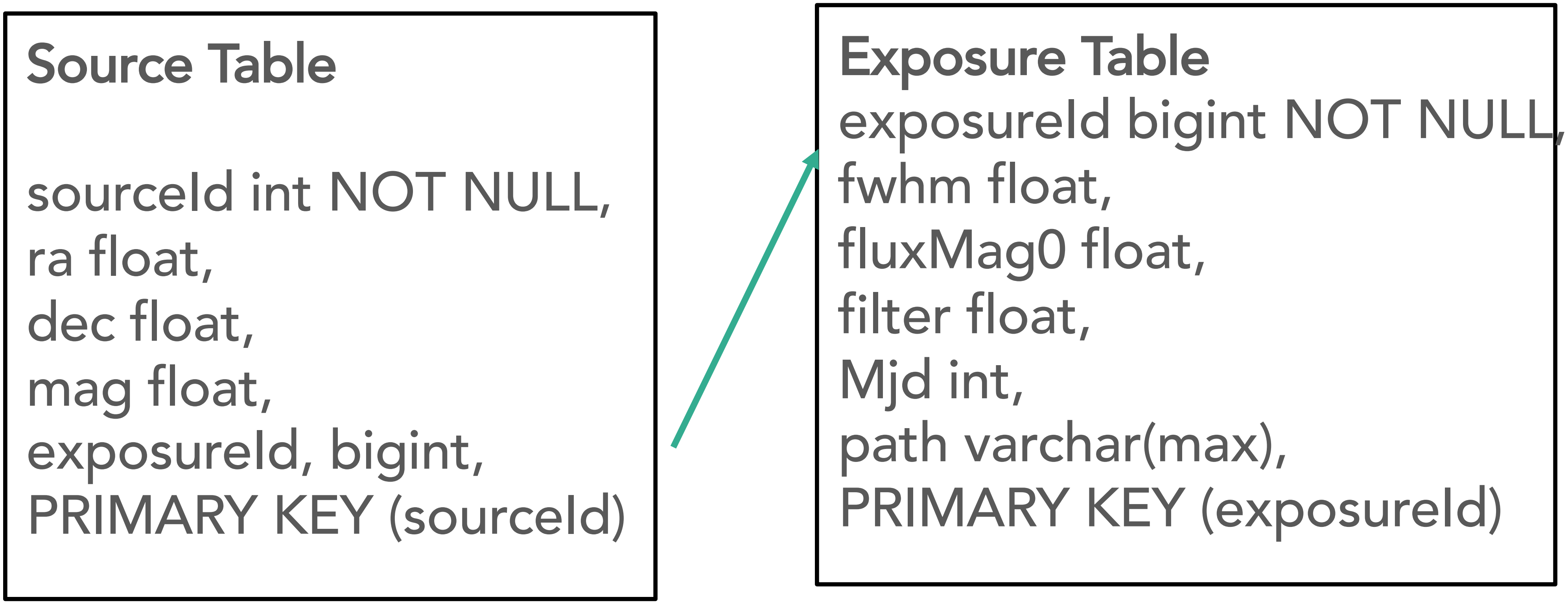


```
|--decam
|--gapon
|--hsc
|  |--calib
|  |--raw
|  |--repo
|  |  |--CFHTLS_W1
|  |  |--COMET2014F3
|  |  |--COSMOS
|  |  |  |--2014-02-02
|  |  |  |--2014-02-05
|  |  |  |--2014-03-26
|  |  |  |  |--00815
|  |  |  |  |  |--HSC-Y
|  |  |  |  |  |  |--HSC-0000628-000.fit
|  |  |  |  |  |  |--HSC-0000636-038.fit
|  |  |  |  |  |  |--HSC-0000662-042.fit
|  |  |  |  |  |  |--HSC-0000670-080.fit
|  |  |  |  |  |  |--  (and 2912 more fil
|  |  |  |--2014-03-27
|  |  |  |--2015-01-17
|  |  |  |--2015-01-18
|  |  |  |--2015-01-19
|  |  |  |--2015-01-20
```

Table (or Relational) Structure

```
Id | taiObs       | expId          |point|dataType| visit    | dateObs      | fra
---|-------------|---------------|-----|--------|---------|------------|----
1  | 2014-02-02 | HSCA90585000 | 763 | OBJECT | 905850 | 2014-02-02 | HSC
2  | 2014-02-05 | HSCA90794800 | 766 | OBJECT | 907948 | 2014-02-05 | HSC
4  | 2014-02-03 | HSCA90679200 | 764 | OBJECT | 906792 | 2014-02-03 | HSC
5  | 2014-02-01 | HSCA90554800 | 762 | OBJECT | 905548 | 2014-02-01 | HSC
7  | 2014-02-04 | HSCA90751400 | 765 | OBJECT | 907514 | 2014-02-04 | HSC
9  | 2014-02-05 | HSCA90806200 | 766 | OBJECT | 908062 | 2014-02-05 | HSC
10 | 2014-02-03 | HSCA90699800 | 764 | OBJECT | 906998 | 2014-02-03 | HSC
11 | 2014-02-01 | HSCA90560000 | 762 | OBJECT | 905600 | 2014-02-01 | HSC
12 | 2014-02-05 | HSCA90794000 | 766 | OBJECT | 907940 | 2014-02-05 | HSC
13 | 2014-02-01 | HSCA90565800 | 762 | OBJECT | 905658 | 2014-02-01 | HSC
14 | 2014-02-01 | HSCA90542000 | 762 | OBJECT | 905420 | 2014-02-01 | HSC
15 | 2014-02-01 | HSCA90563200 | 762 | OBJECT | 905632 | 2014-02-01 | HSC
16 | 2014-02-03 | HSCA90655200 | 764 | OBJECT | 906552 | 2014-02-03 | HSC
17 | 2014-02-04 | HSCA90761200 | 765 | OBJECT | 907612 | 2014-02-04 | HSC
19 | 2014-02-03 | HSCA90664400 | 764 | OBJECT | 906644 | 2014-02-03 | HSC
20 | 2014-02-03 | HSCA90702600 | 764 | OBJECT | 907026 | 2014-02-03 | HSC
Etc…
```

Table (or Relational) Model Example



Source Table

sourceId int NOT NULL,
ra float,
dec float,
mag float,
exposureId, bigint,
PRIMARY KEY (sourceId)

Exposure Table

exposureId bigint NOT NULL,
fwhm float,
fluxMag0 float,
filter float,
Mjd int,
path varchar(max),
PRIMARY KEY (exposureId)

Table (or Relational) Structure

```
SourceId | psfMag | filter |      fluxMag0
|---------|--------|--------|------------------
|1         |  21.1  |  2       |  5721049381928.0
|2         |  20.1  |  2       |  5721049381928.0
|4         |  19.3  |  2       |  5721049381928.0
|5         |  18.3  |  2       |  5721049381928.0
|7         |  19.6  |  2       |  5721049381928.0
|9         |  20.1  |  2       |  5721049381928.0
|10        |  20.1  |  2       |  5721049381928.0
```

```
SELECT s.sourceId, s.psfMag, e.filter, e.fluxMag0
FROM Source s
INNER JOIN Exposure e
ON s.exposureId = e.exposureId
```

How many friends does Alice have?

Friendships

PersonId1
PersonId2
BeginDate

Person

PersonId
FirstName
LastName

```
SELECT a.Person2
FROM Friendships a
INNER JOIN Person p
ON p.personId = a.personId1
WHERE p.FirstName = ’Alice’

UNION ALL

SELECT a.Person1
FROM Friendships b
INNER JOIN Person p
ON p.personId = a.personId2
WHERE p.FirstName = ’Alice’
```

How many *friends of friends* does Alice have??

Friendships

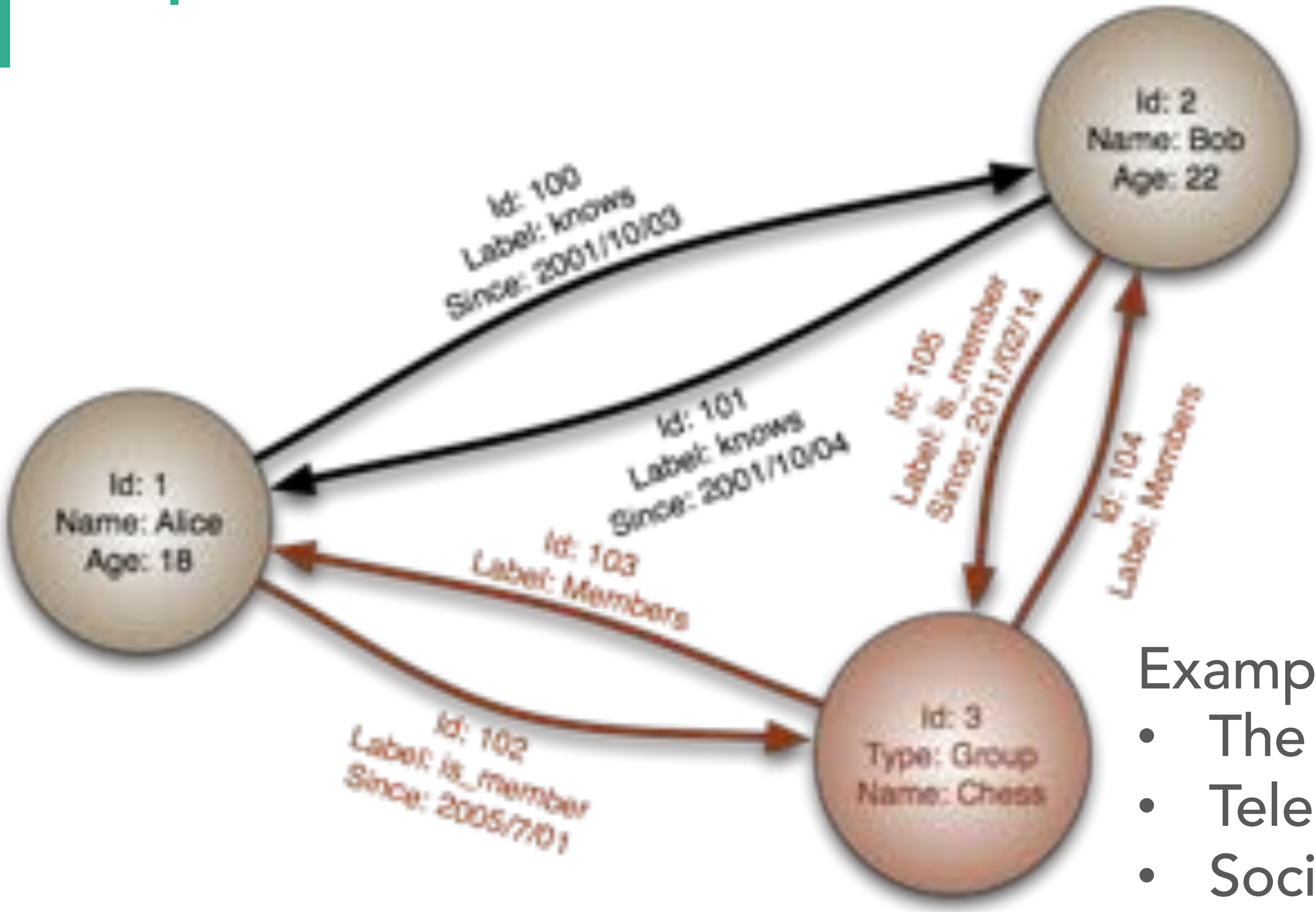
PersonId1
PersonId2
BeginDate

Person

PersonId
FirstName
LastName

```
SELECT a.Person2
FROM Friendships a
INNER JOIN Person p
ON p.personId = a.personId1
WHERE p.FirstName = ’Alice’

UNION ALL

SELECT a.Person1
FROM Friendships b
INNER JOIN Person p
ON p.personId = a.personId2
WHERE p.FirstName = ’Alice’
```

Graph Structure



Examples:

- The web
- Telecom
- Social Networks

Image credit: wikipedia

Data models are defined by

- Structures
  - Tree
  - Relational (Tables)
  - Graphs (*nodes and edges)*
  - Keys and values

- Constraints
  - Examples:
    - File can't be in two folders.
    - All rows have same number of columnd
- Operations

SQL as Relational Algebra

Operations on table implemented by Relational Database Management Systems (RDMS)

Relational Algebra

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* (×) Allows us to combine two relations.
    - Inner join, left join, right join, outer join
  - *Set-difference* (−) Tuples in A, but not in B
  - *Union* (∪) Tuples in A or B
- Additional operations:
  - Intersection, *join*, division, renaming
- Since each operation returns a relation, operations can be *composed*! (Algebra is "closed".)

Adapted from Database Management Systems, R. Ramakrishnan and J. Gehrke

Logical Join Types

Aggregate Operators and GROUP BY

- COUNT (*)
- COUNT ([DISTINCT] A)
- SUM (A)
- AVG (A)
- MAX (A)
- MIN (A)

```
SELECT plate, COUNT(id)
FROM SpecObjAll
GROUP BY plate
```

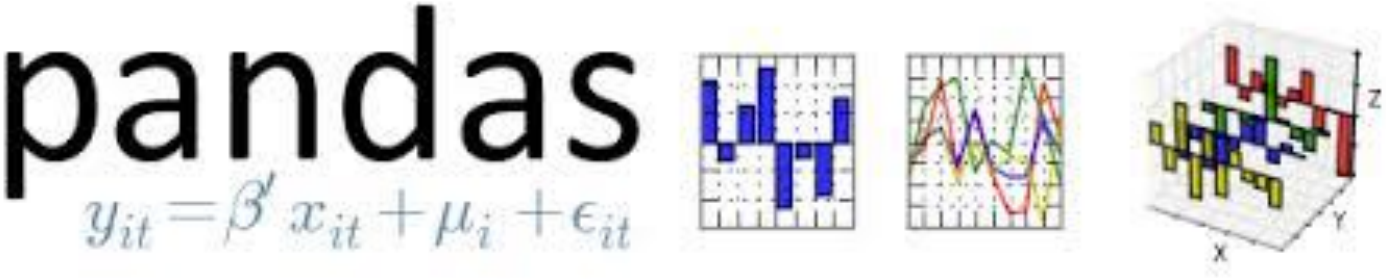
Alternative implementations of Relational Algebra

Interlude to talk about some tools

SQL Databases are not necessary to get Relational Algebra

Wes McKinney's personal story on birth of pandas:

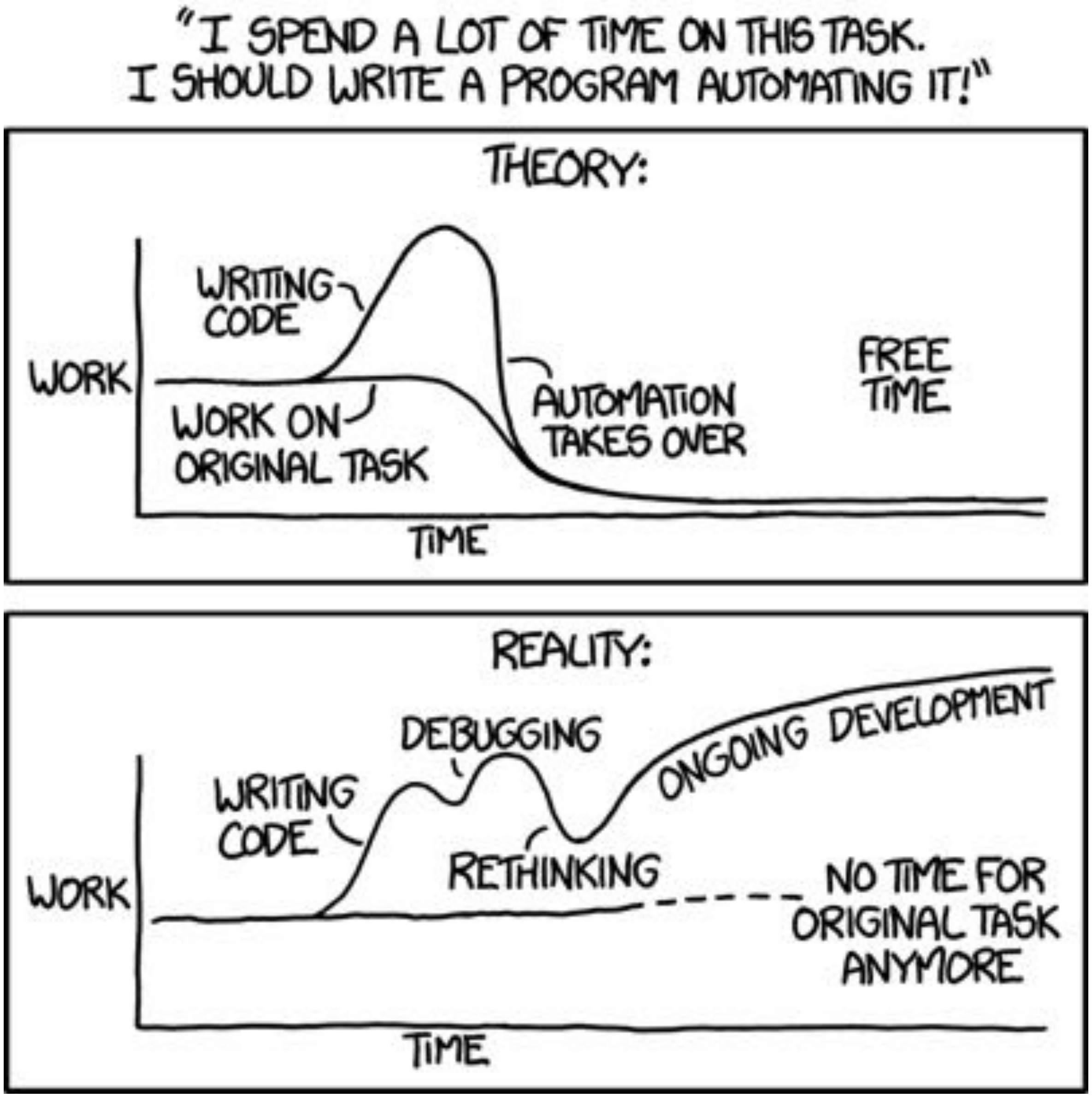
youtube.com/watch?v=kHdkFyGCxiY



You can SOMETIMES use pandas if the dataset doesn’t fit into memory

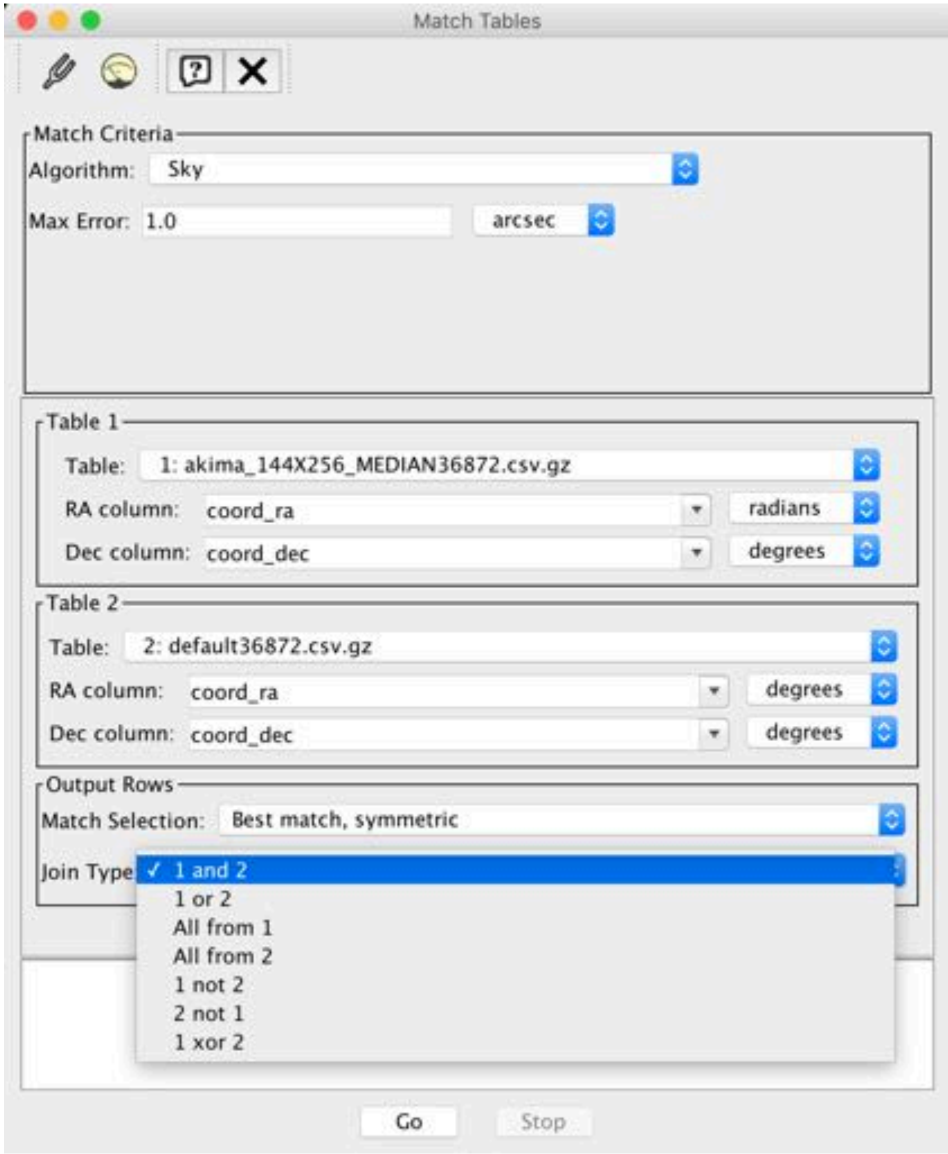
```
dc = pd.read_csv(filenameDeepCoadd, compression='gzip',
                 header=None, index_col=0)

sources = pd.read_csv(filenameDeepSource,
                      compression='gzip', header=None,
                      chunksize=100000, index_col=0)

for i, chunk in enumerate(sources):
    # Now you can join the chunk after some prep:
    joined = pd.merge(chunk, dc, left_on='deepCoaddId',
                      right_index=True)
  # And so on
```

Beware of over-engineering



TOPCAT is written specifically for astronomers by an astronomer

GUI tool for relational algebra on tables
+ fits table I/O
+ theta joins ON RA/DEC



Match Tables

Match Criteria
Algorithm: Sky
Max Error: 1.0 arcsec

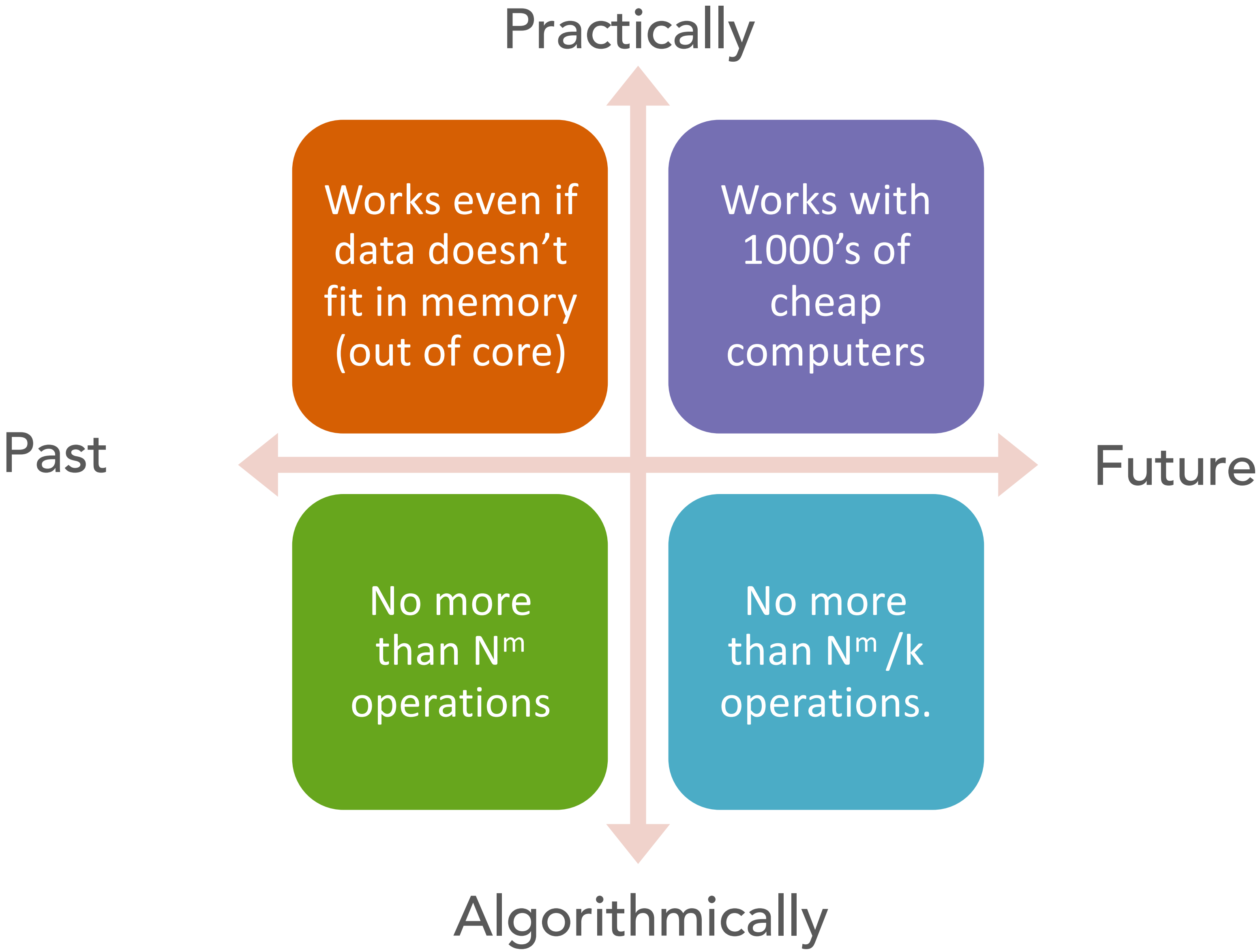
Table 1
Table: 1: akima_144X256_MEDIAN36872.csv.gz
RA column: coord_ra radians
Dec column: coord_dec degrees

Table 2
Table: 2: default36872.csv.gz
RA column: coord_ra degrees
Dec column: coord_dec degrees

Output Rows
Match Selection: Best match, symmetric
Join Type: ✓ 1 and 2

Indexes (and Spatial Indexes)

What do we mean by scalability?



Practically

Past

Future

Algorithmically

Works even if data doesn’t fit in memory (out of core)

Works with 1000’s of cheap computers

No more than N^m operations

No more than N^m/k operations.

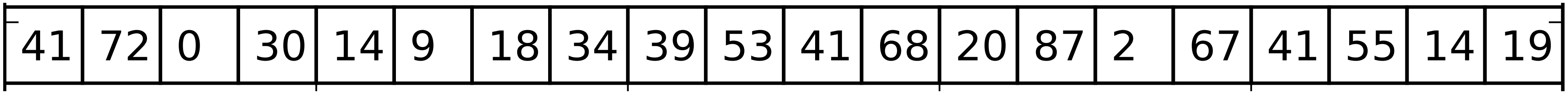
Scalability

- Practically:
  - Before: Works even if data doesn't fit in memory (out of core)
  - Now: Can we utilize 1000's of cheap computers
- Algorithmically :
  - For N data items:
    - Before: You can do no more than N^m operations
    - Now: You can do no more than N^m /k operations
    - Soon: You can do no more than N*log(N) operations (one pass – streaming)

Example: search for records where value = 41

N = 20

How many comparisons must we make?



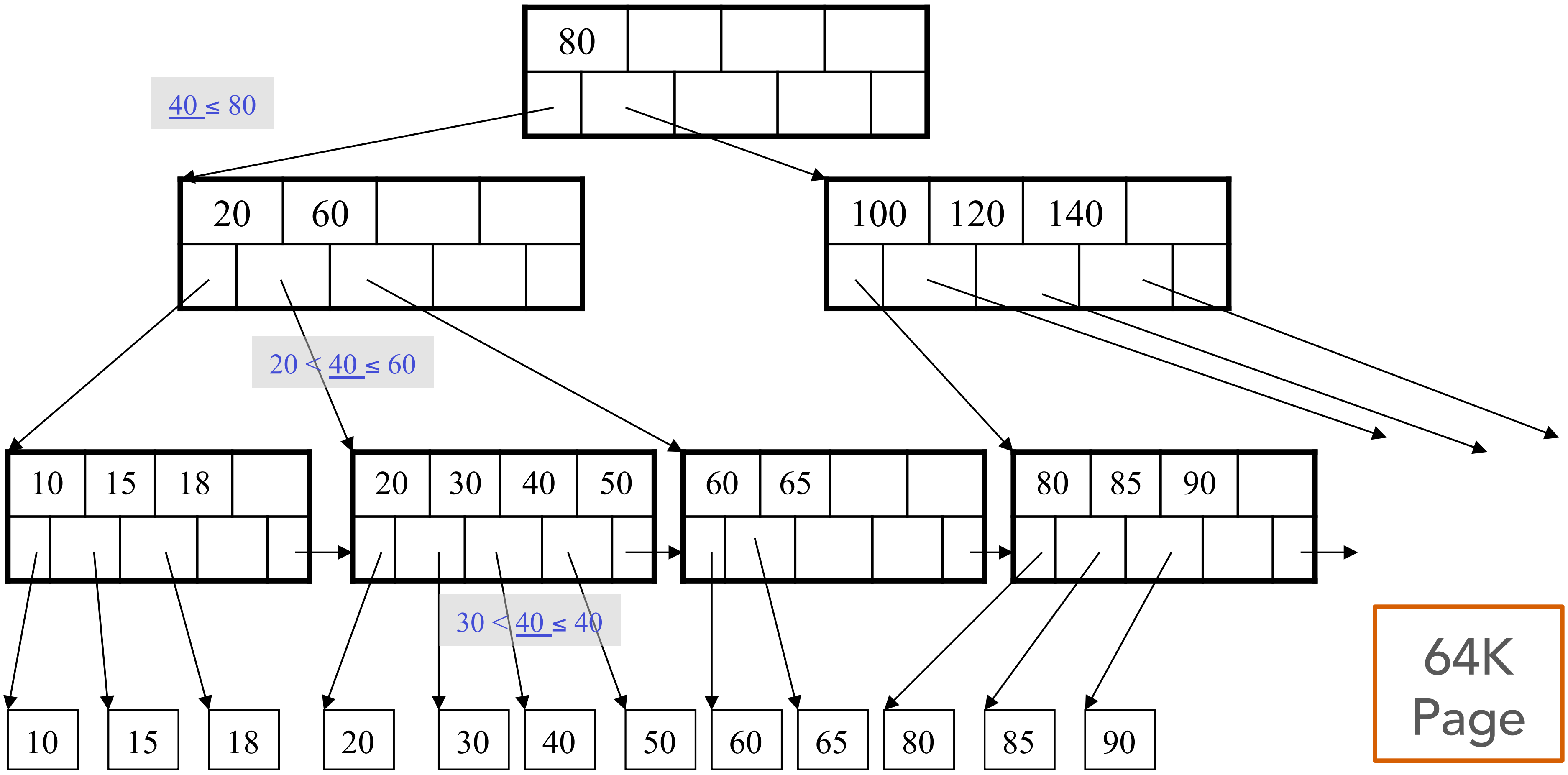
41	72	0	30	14	9	18	34	39	53	41	68	20	87	2	67	41	55	14	19

Example: search for records where value = 41

N = 20
How many comparisons must we make?

0	2	9	14	14	18	19	20	30	34	39	41	41	41	53	55	67	68	72	87

B+ Tree Index



40 ≤ 80

20 < 40 ≤ 60

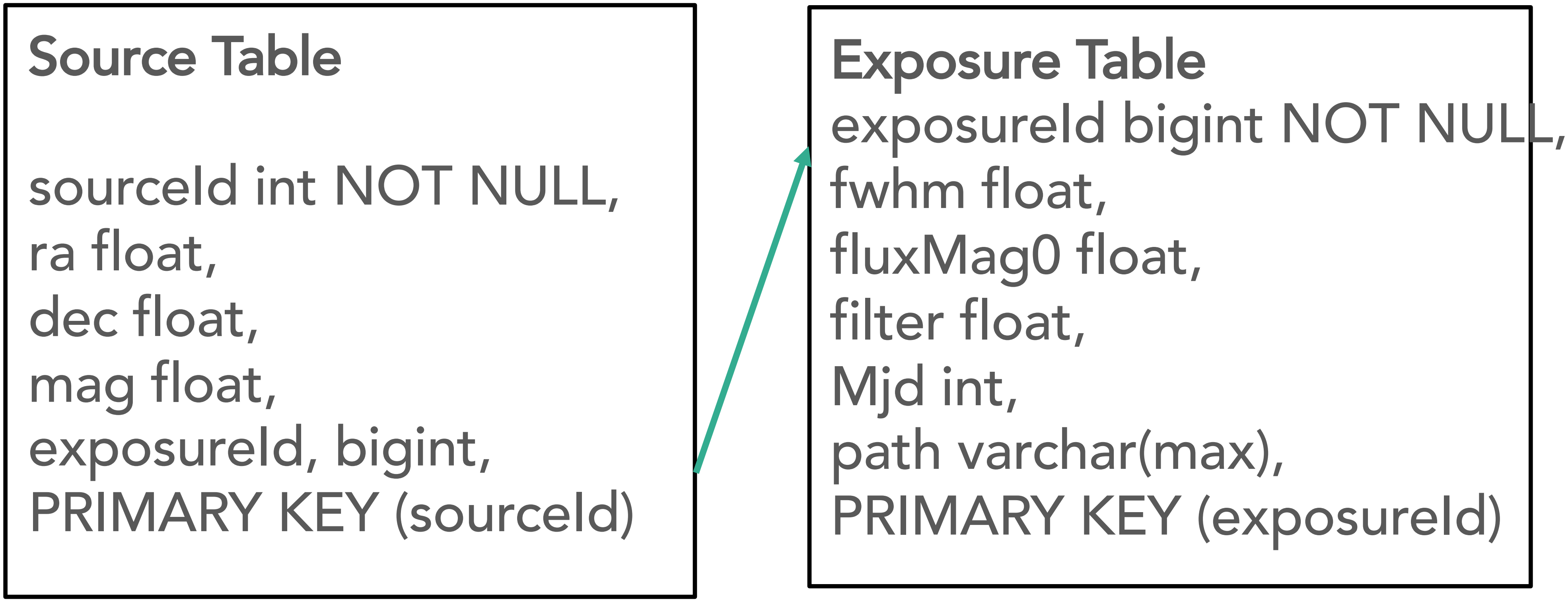
30 < 40 ≤ 40

64K Page

Image credit: UW CSE

Relational databases excel at finding record sets within large datasets

What are good choices of columns for indexing?



Source Table

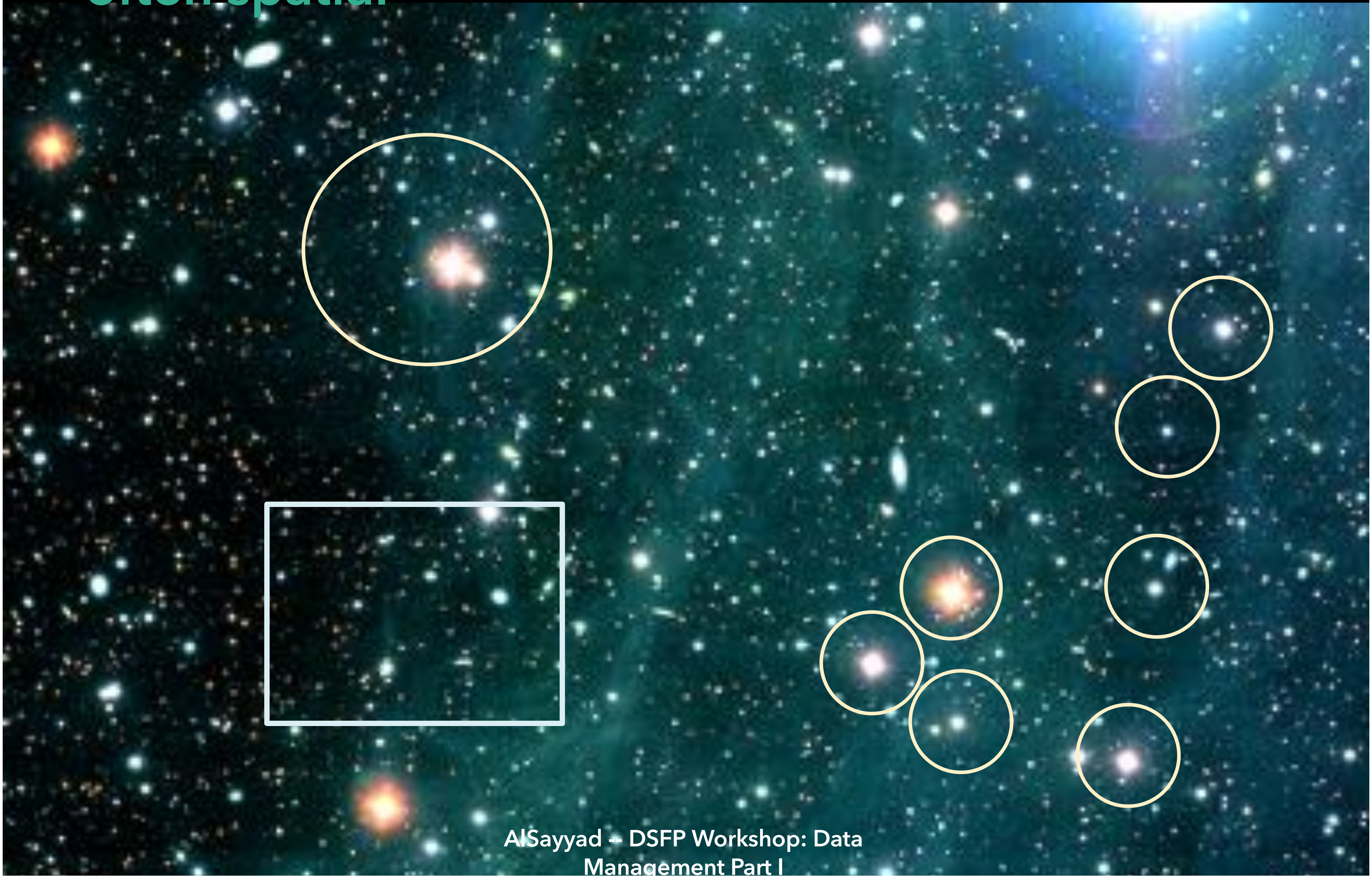
sourceId int NOT NULL,
ra float,
dec float,
mag float,
exposureId, bigint,
PRIMARY KEY (sourceId)

Exposure Table

exposureId bigint NOT NULL,
fwhm float,
fluxMag0 float,
filter float,
Mjd int,
path varchar(max),
PRIMARY KEY (exposureId)

```
CREATE INDEX idx_exposureId ON Source(exposureId);
```

In astronomical applications those seeks are often spatial



But coordinates are usually stored as RA, DEC

What do we index on

Many databases ship with Geographic Information System (GIS) support or have plugins

- Select a spherical model of the Earth
- Map:
  - R.A. --> Longitude
  - Dec. --> Latitude

Microsoft® SQL Server® 2008

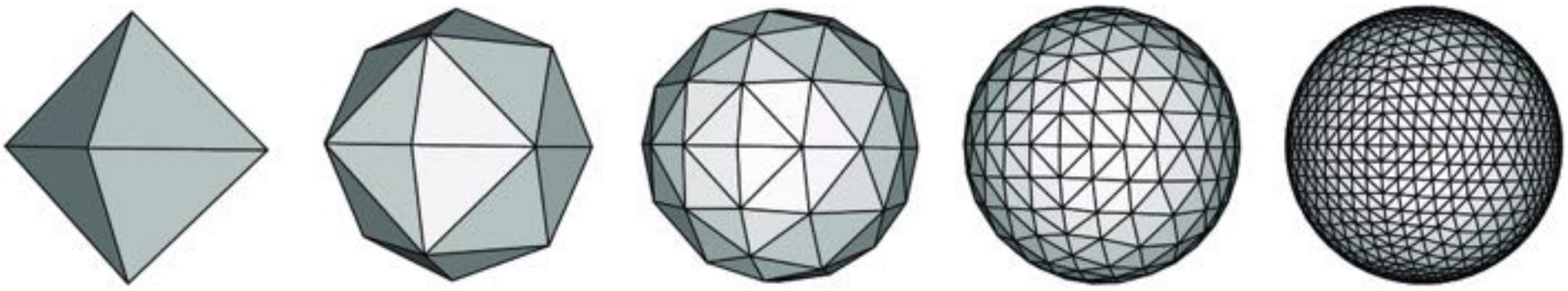
PostGIS

Spatial PostgreSQL

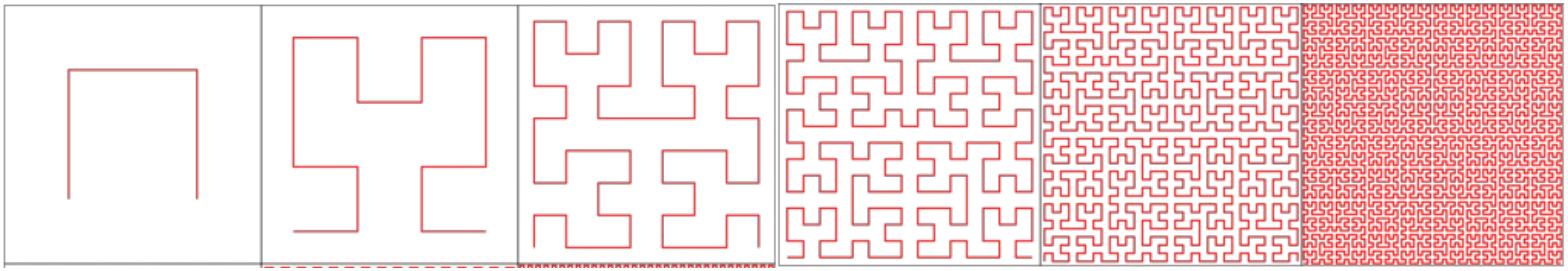
OpenGIS®

See en.wikipedia.org/wiki/Spatial_database for a long list

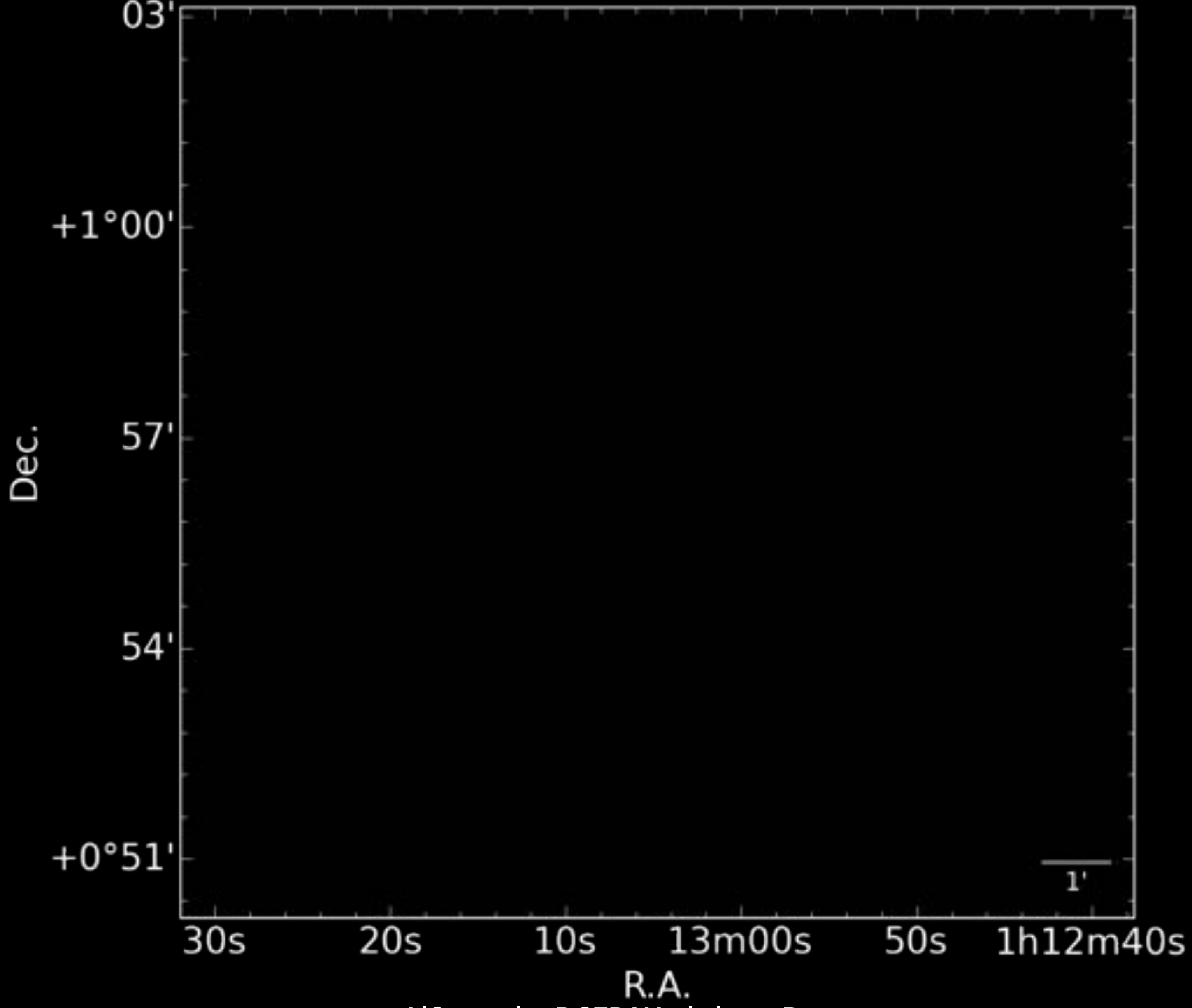
Astronomers are fond of Hierarchical Triangular Mesh

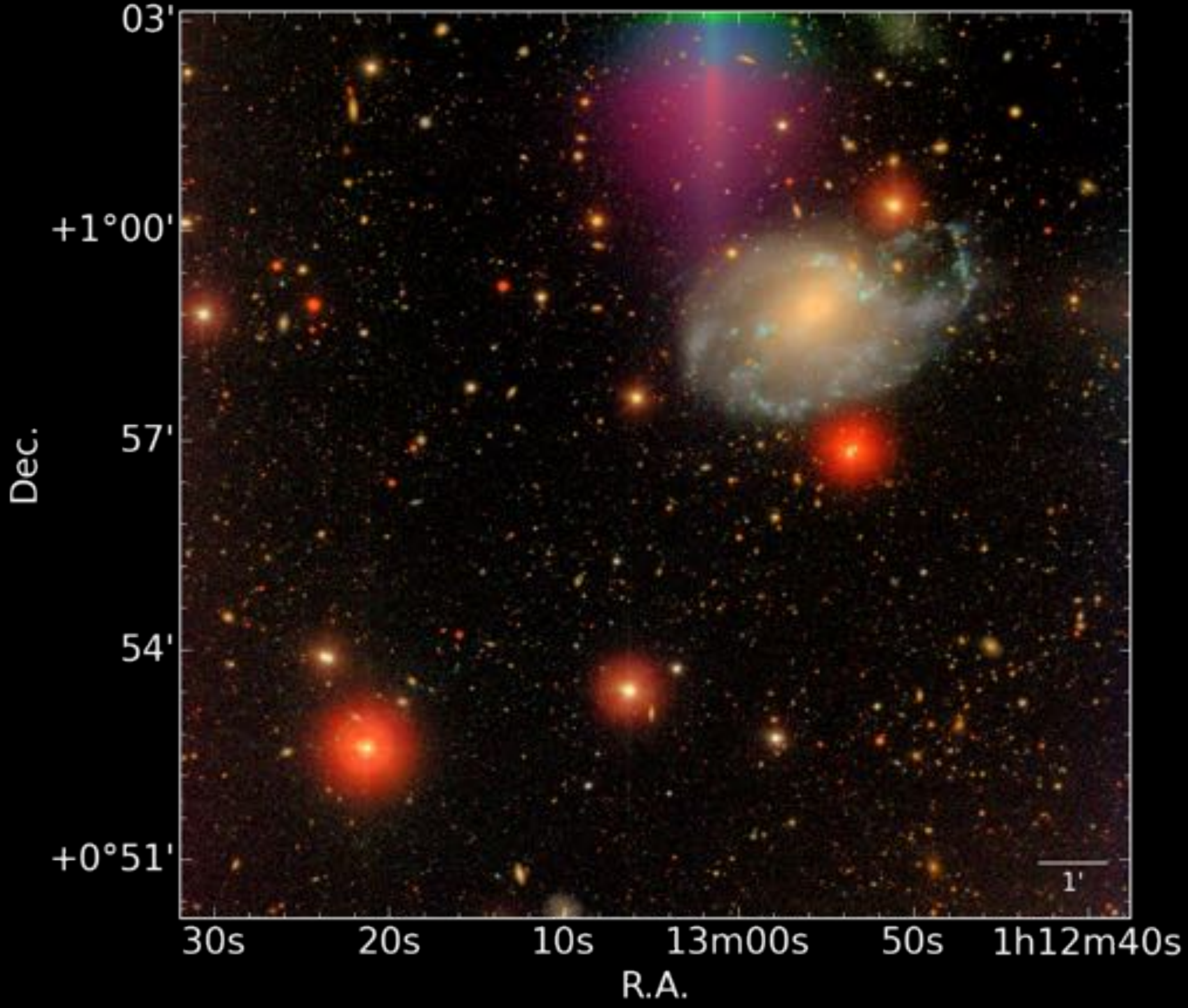


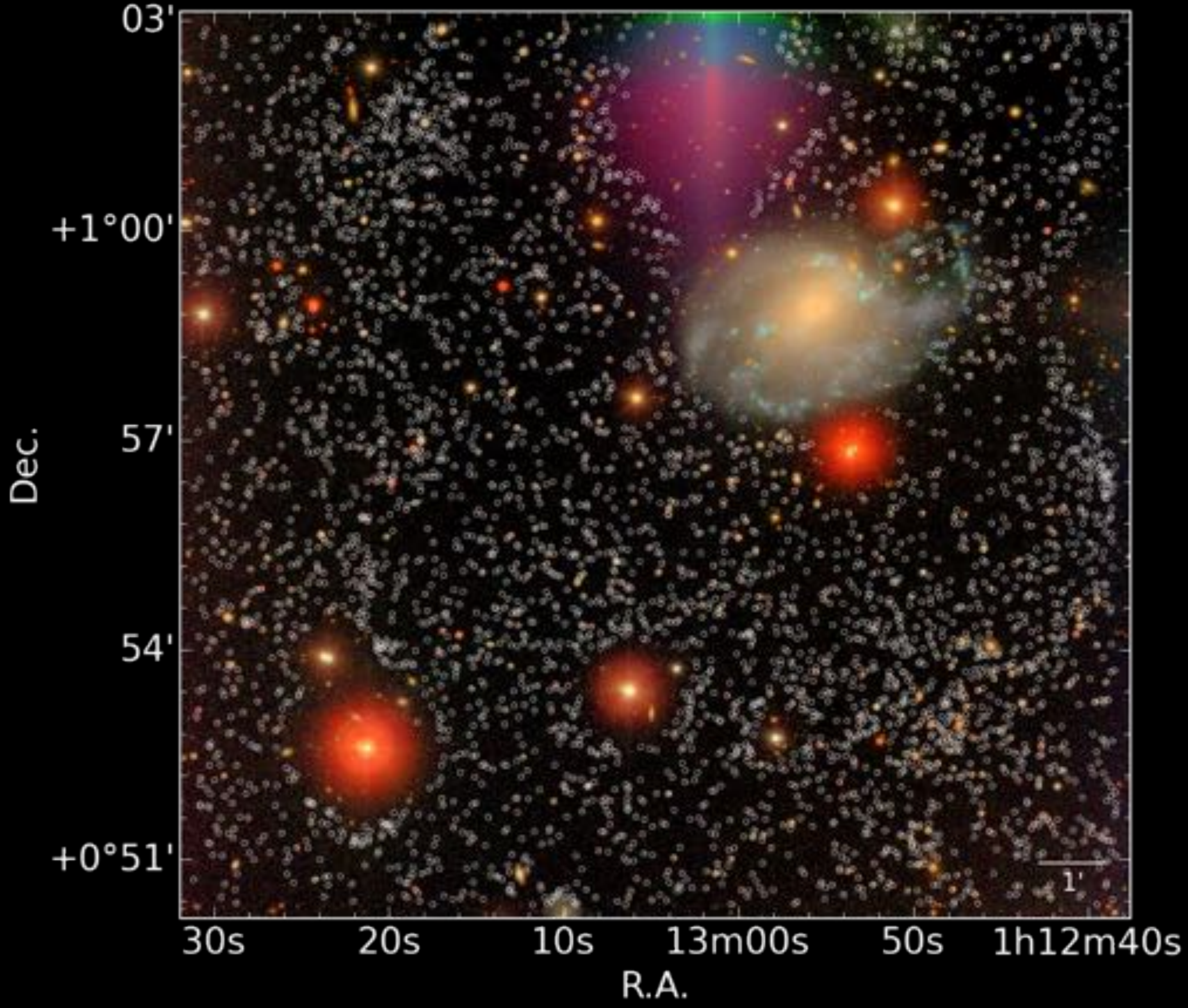
Reduce 2 dimensions to 1 with space filling curve

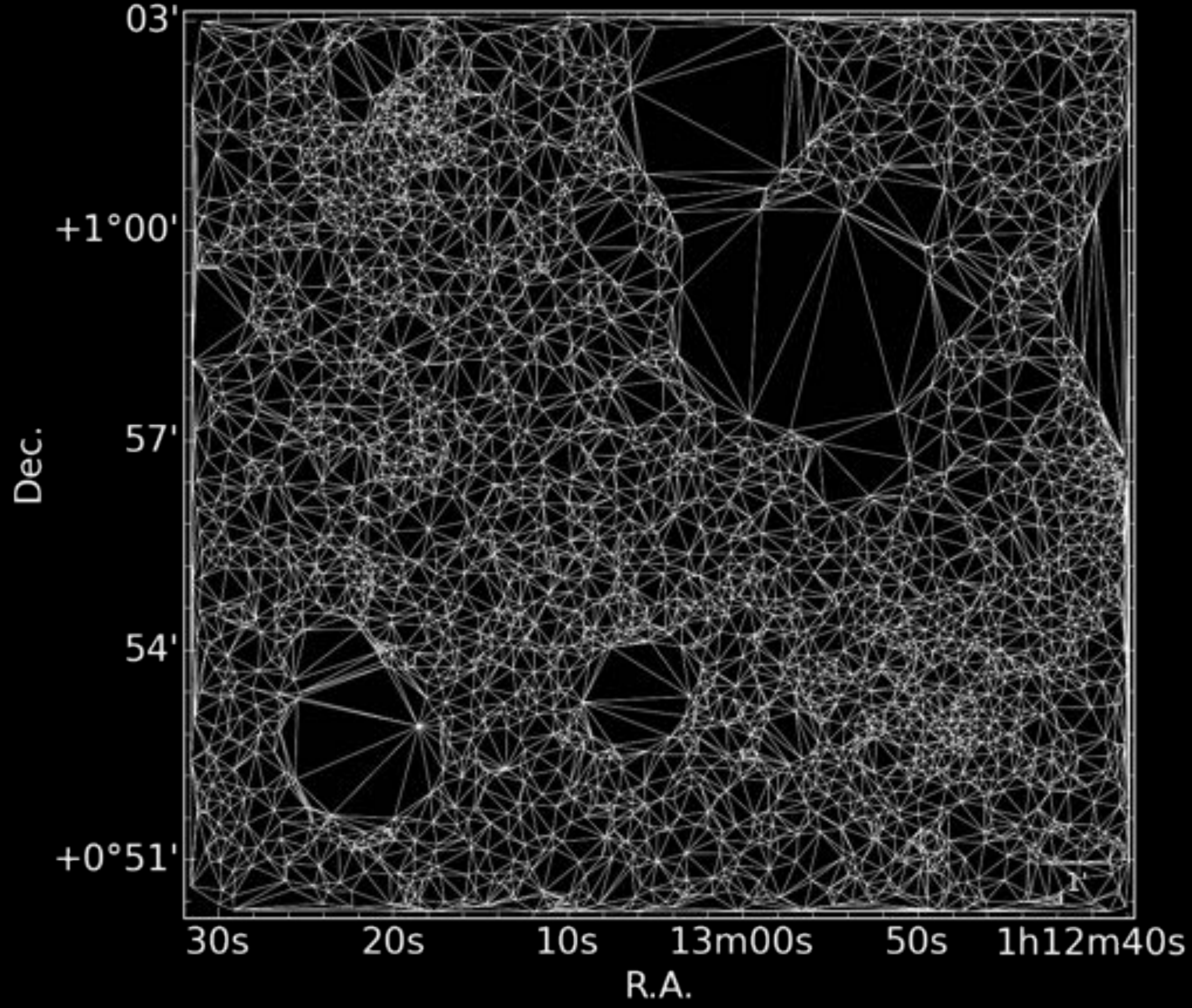


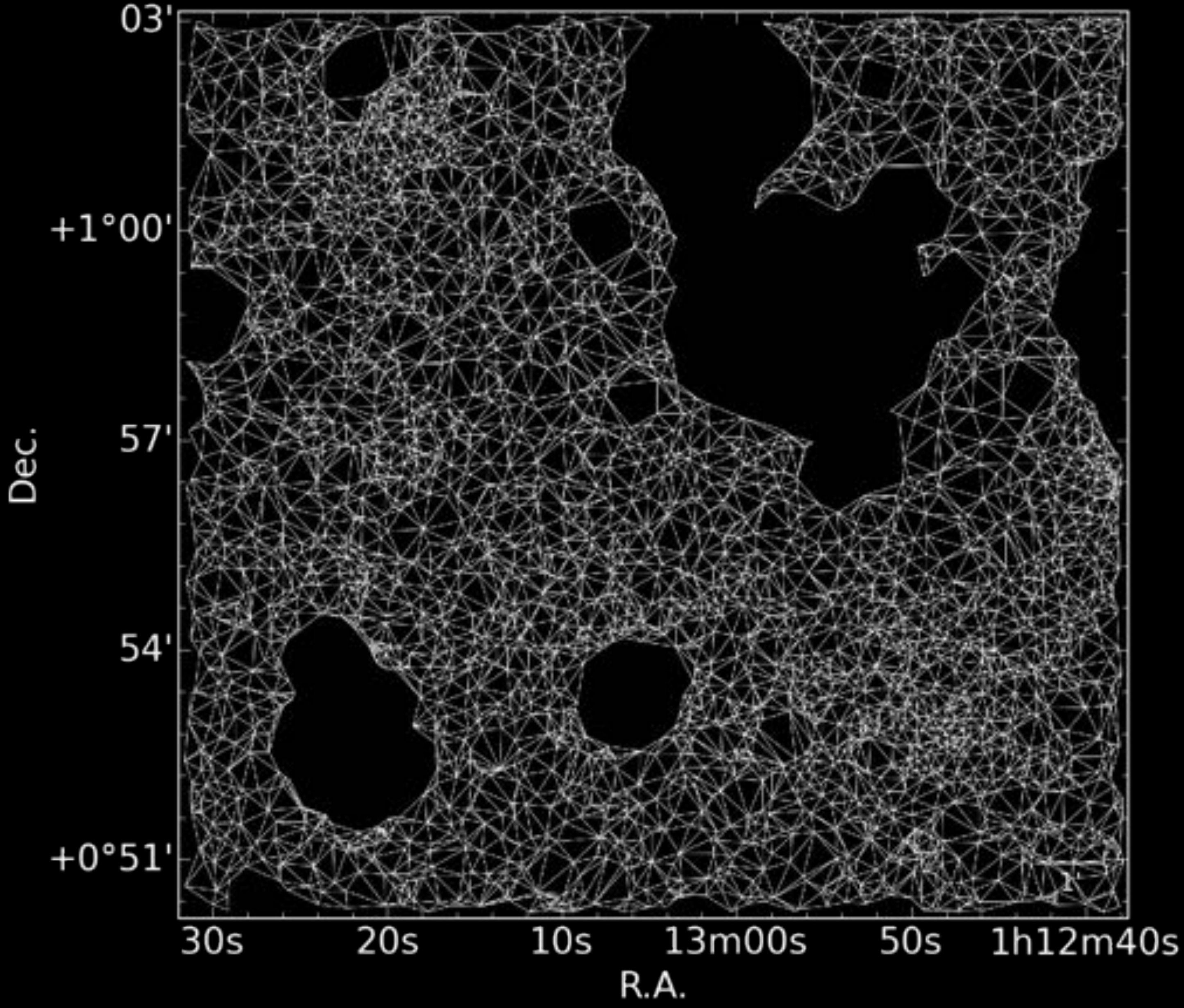
Background Information for the challenge problem

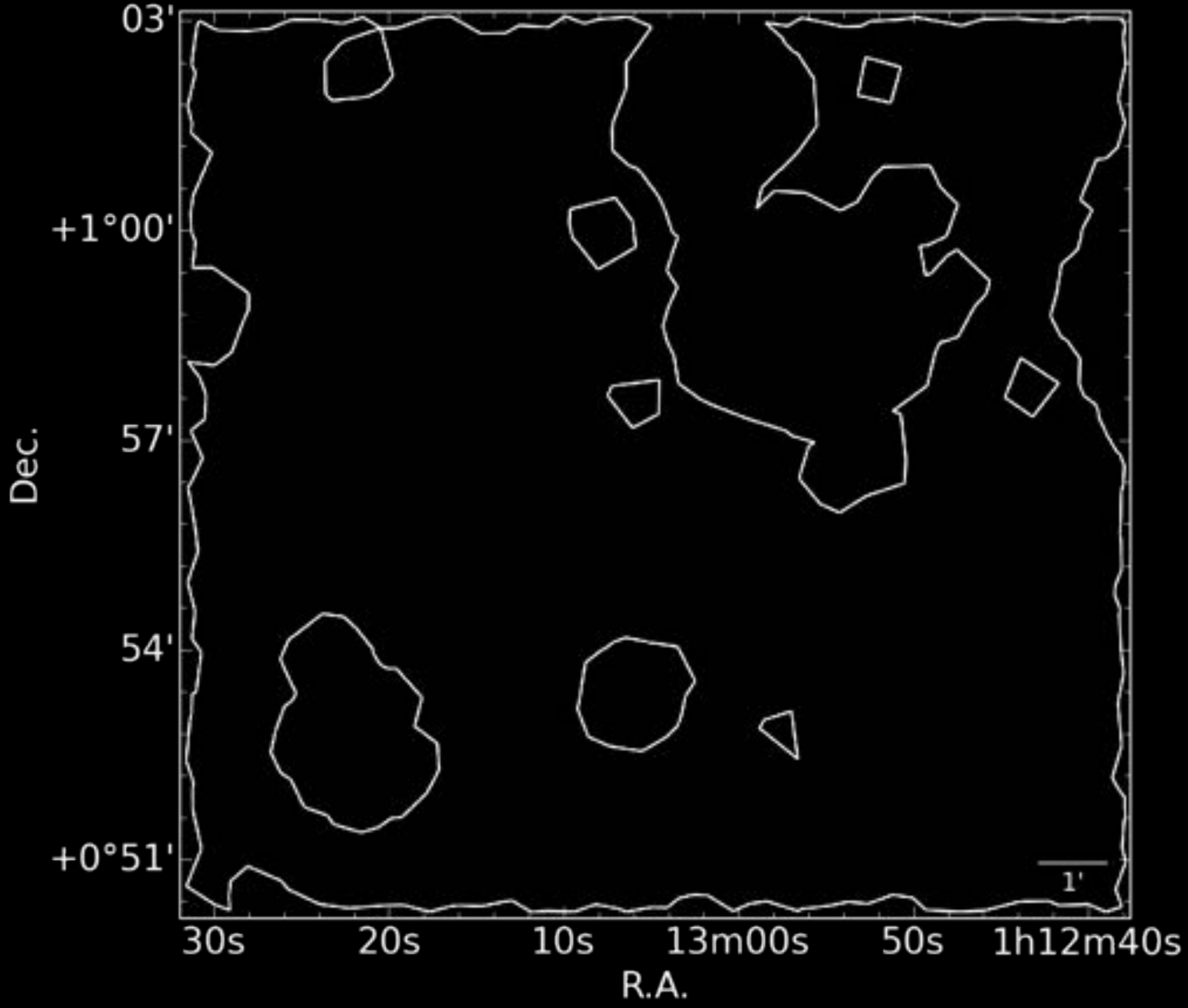



03'
+1°00'
57'
54'
+0°51'
Dec.
30s
20s
10s
13m00s
50s
1h12m40s
R.A.
1'


03'
+1°00'
57'
54'
+0°51'
Dec.
30s
20s
10s
13m00s
50s
1h12m40s
R.A.
1'


03'
+1°00'
57'
54'
+0°51'
Dec.
30s
20s
10s
13m00s
50s
1h12m40s
R.A.
1'


03'
+1°00'
57'
54'
+0°51'
Dec.
30s
20s
10s
13m00s
50s
1h12m40s
R.A.
1'


03'
+1°00'
57'
54'
+0°51'
Dec.
30s
20s
10s
13m00s
50s
1h12m40s
R.A.
1'