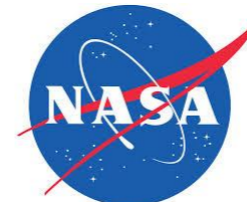


Leveraging CAOM to Implement SIA2 and SSA on a Billion* Row Table

Walter Landry



Caltech



Common Archive Object Model

- "A general purpose data model for use as the core data model of an astronomical data center"
- CADC has based their whole infrastructure around it.
- MAST uses it for operational data.
- INAF is moving towards it
- IRSA has decided to adopt it for all of their image and spectra (not catalogs).

CAOM Maps Well to a Series of Tables

- **Observation**
 - High level information: telescope, PI, instrument, ...
- **Plane**
 - Physics information: geometry, energy bounds, time bounds, ...
- **Artifact**
 - File level information: MIME type, URI, ...

Row ID's

- Every **Plane** belongs to an **Observation**, and every **Artifact** belongs to a **Plane**.
- Every row of every table has a globally unique ID (==UUID?).

Other CAOM Tables

- There are two other tables, but we have not found them useful.
- For example, they are not required to implement SIA2 or SSA.

CAOM is Complete

- ObsCore is too simple of a model to really capture all of our data.
- CAOM is, if anything, too rich.
 - Geolocation for balloons?
- There are some keywords available for mission-specific data
 - Keywords field is plain text.
 - We structure it as a JSON object.

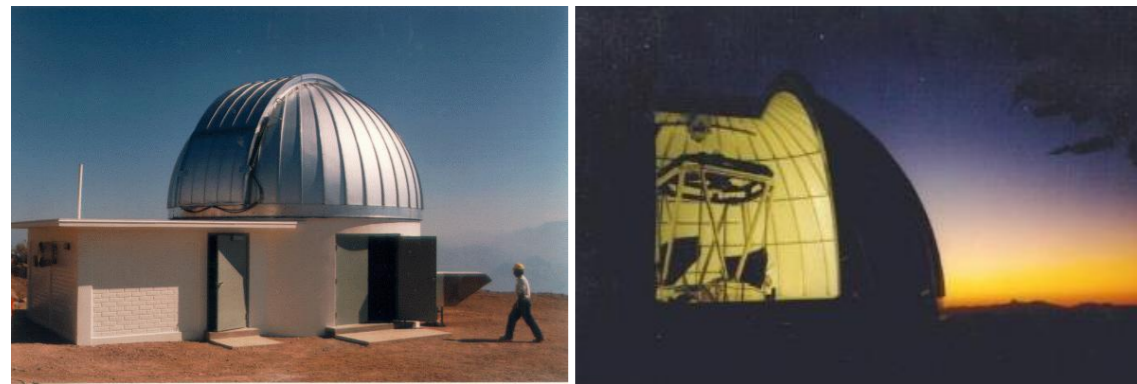
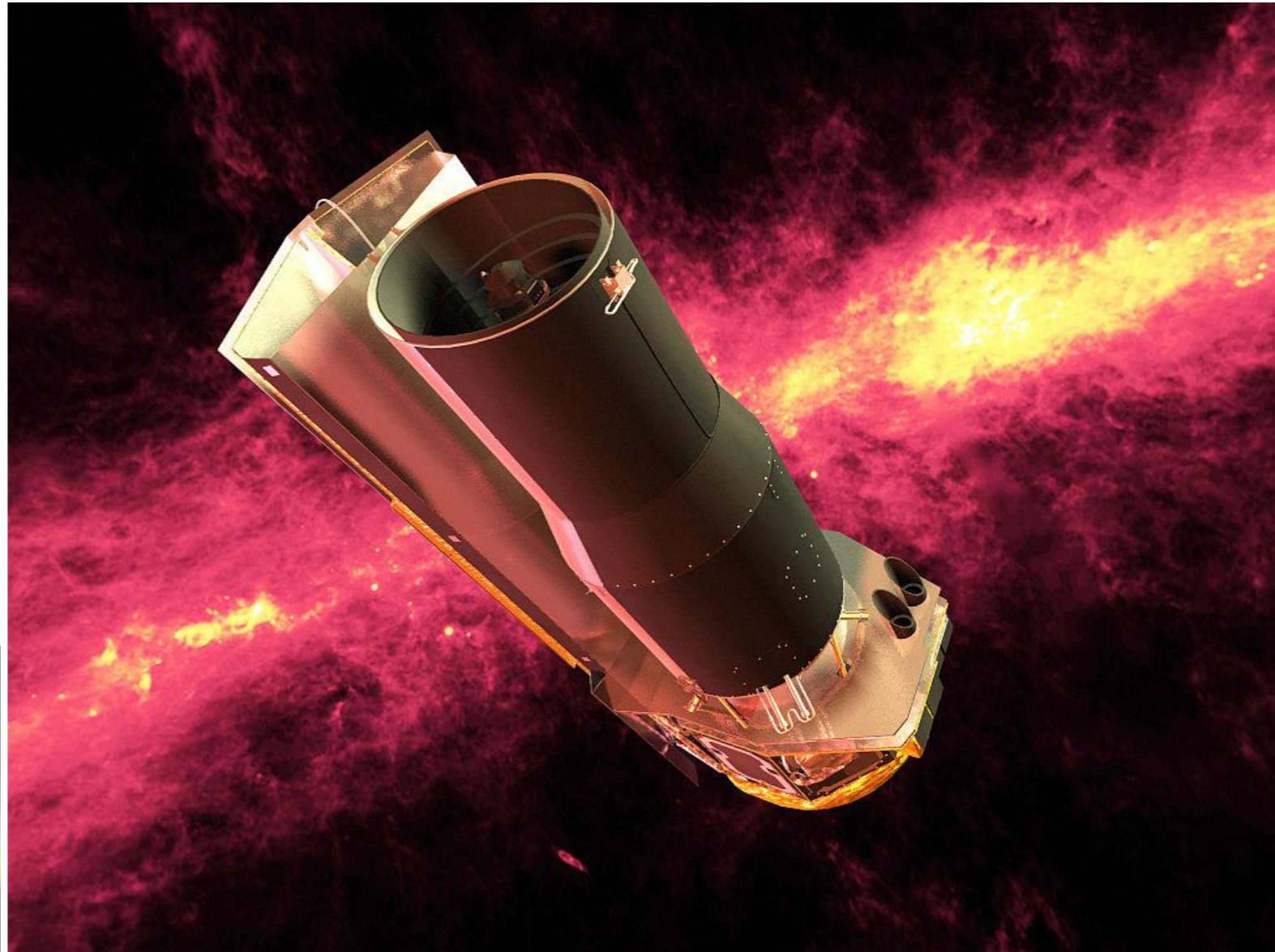
```
{ "reqmode" : [ "IracMap" ] }
```

CAOM is Quite Complicated

- We needed contributions from almost everyone at IRSA to get this working.
- Still fixing some bugs in our interpretation
- Mostly stable?

CAOM at

- Operational tables have WISE, 2MASS, and Spitzer (SHA)



Spitzer Heritage Archive is HUGE

- 250 million files
 - A single SHA observation can yield 80,000 files.
 - Searching a relatively small cone yielded 543,098 results.

File Duplication

- Some of Spitzer's calibration files are valid for a whole campaign.
- This means that a single file can be associated with multiple observations.
- Datalink?
 - We tried something like that (tables link to other tables)
 - Performance was terrible
 - Ended up duplicating rows => 750 million

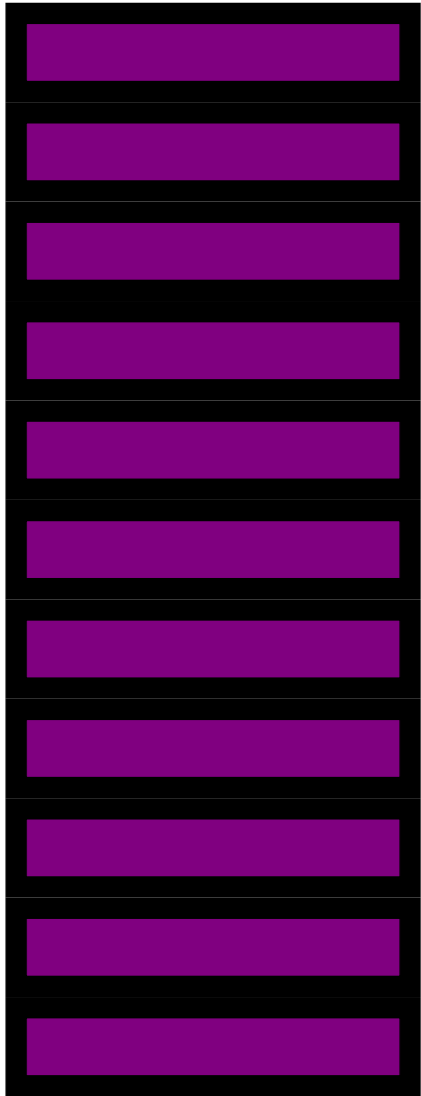
Optimizing Search

- Consider a typical search by an archive user
 - Cone search on **Plane**
 - Join against **Observation** and **Artifact** for other details.
- What is the best way to order the records on disk?

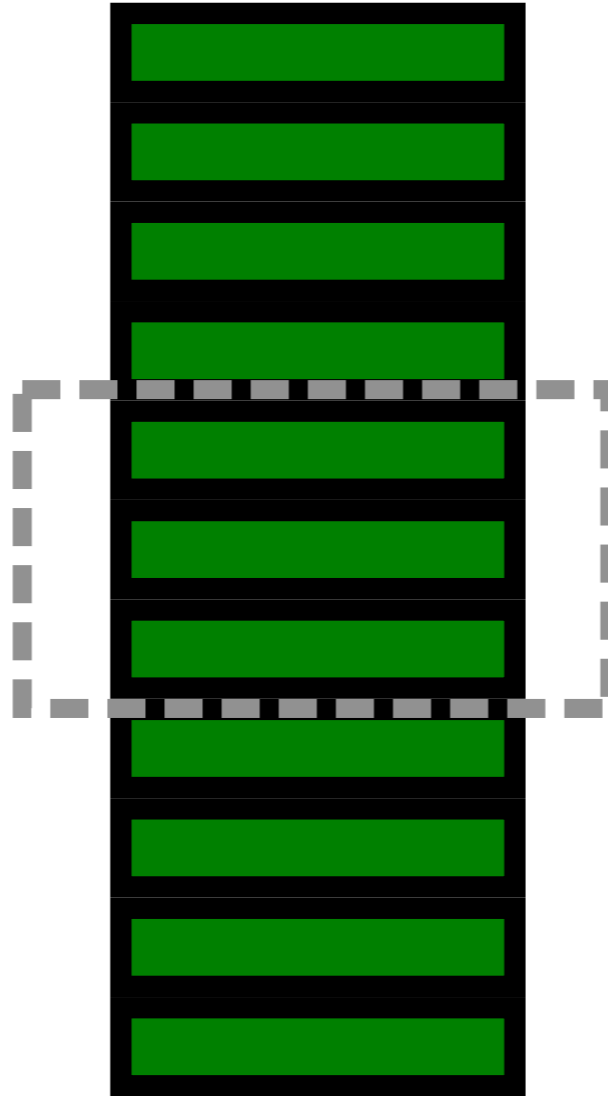
Ordering and Indexing

- Order and index **Plane** by a geometric index
 - PostGIS makes this easy for us
 - Also have an index on the ID.
- To make the joins with **Observation** and **Plane** fast, we need to index them by their ID.
- A natural thing to do is to then order **Observation** by it's ID, and **Artifact** by the **Plane** ID.
- What is the performance of this structure?

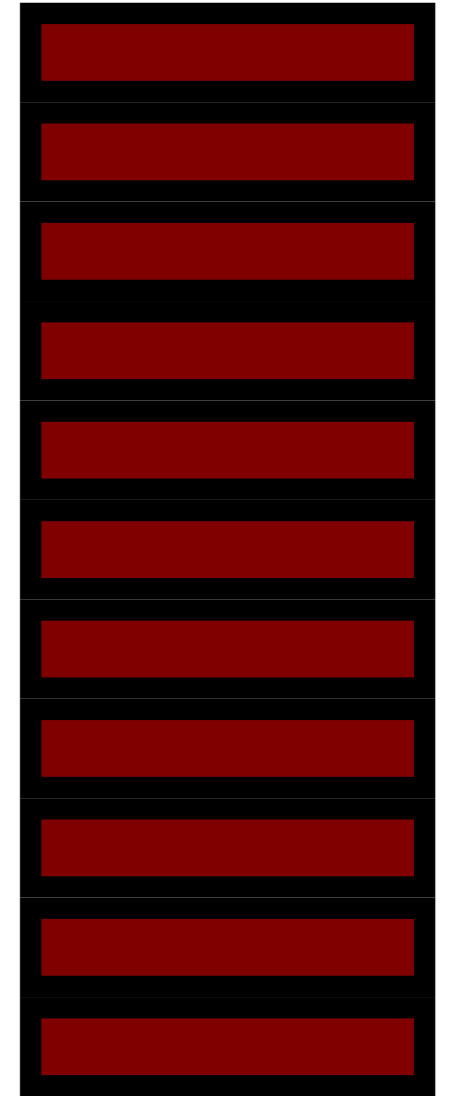
Fetch **Plane** Results



Observation



Plane

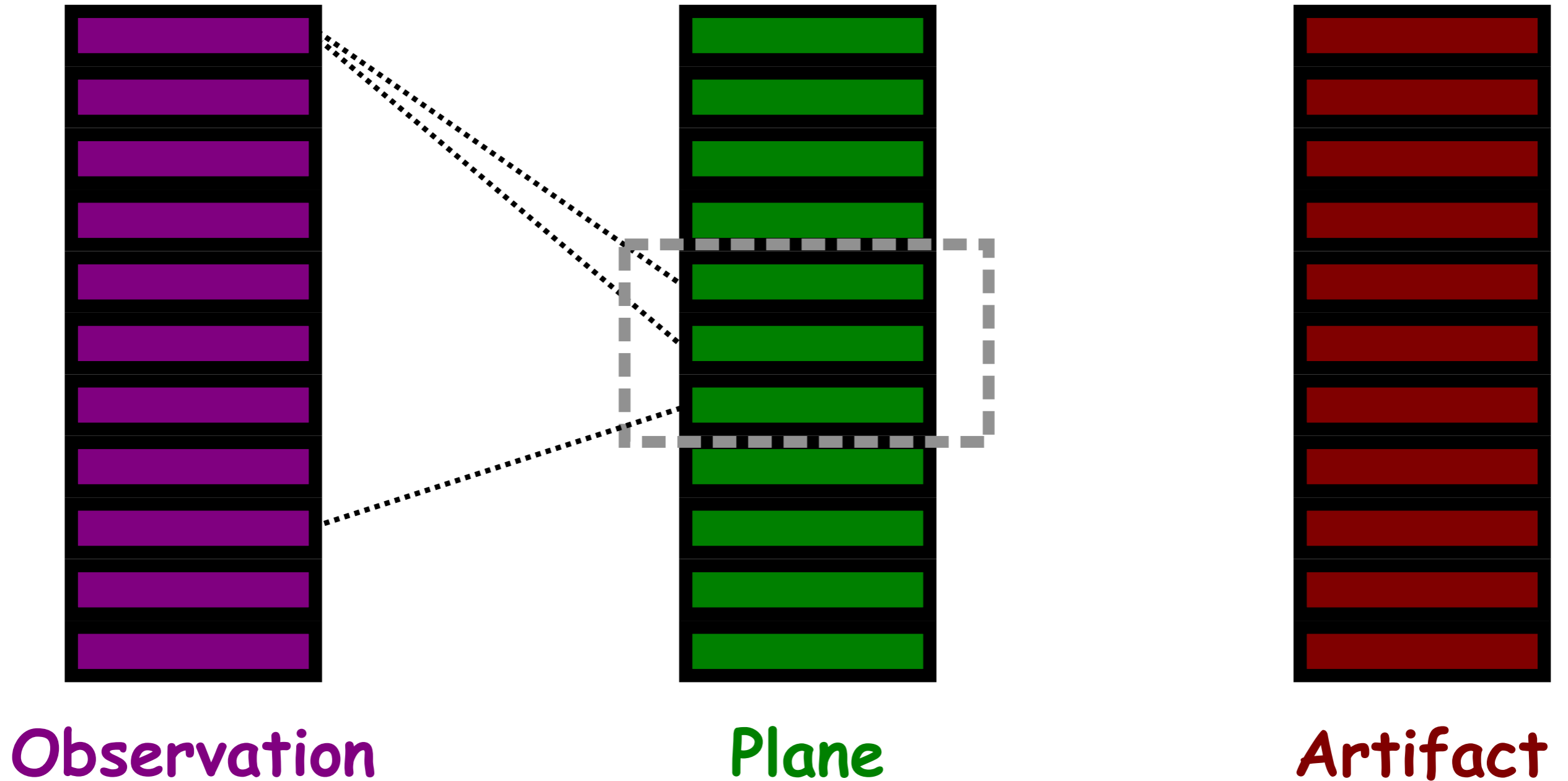


Artifact

Fetching **Plane** Records is Fast

- The rows that are close on the sky are close on disk.

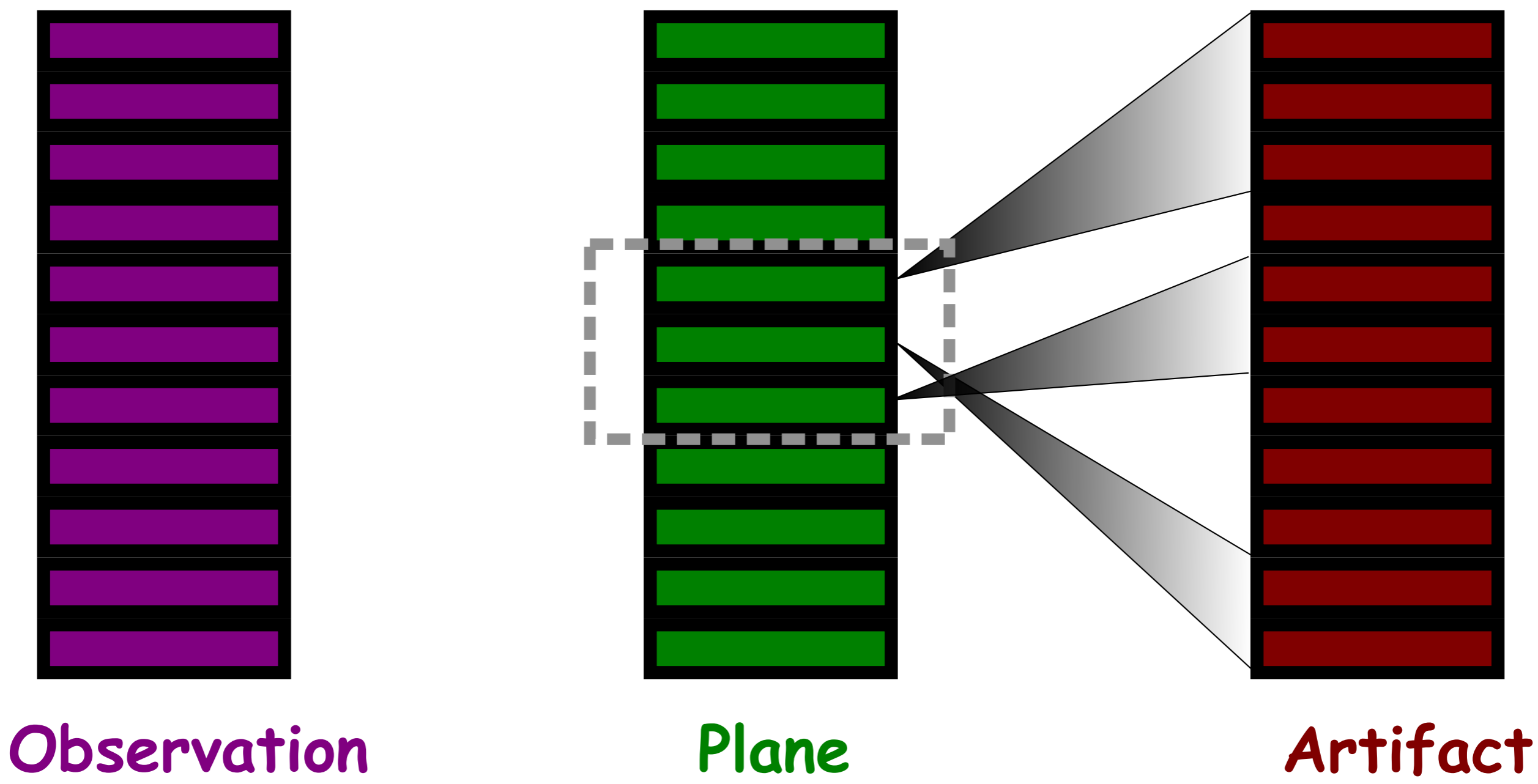
Fetch **Observation** Results



Fetching **Observation** Rows is Terrible

- Ordering by UUID's randomizes disk location.
- In practice, it is not too bad since there are typically many **Plane** rows for each **Observation** row.

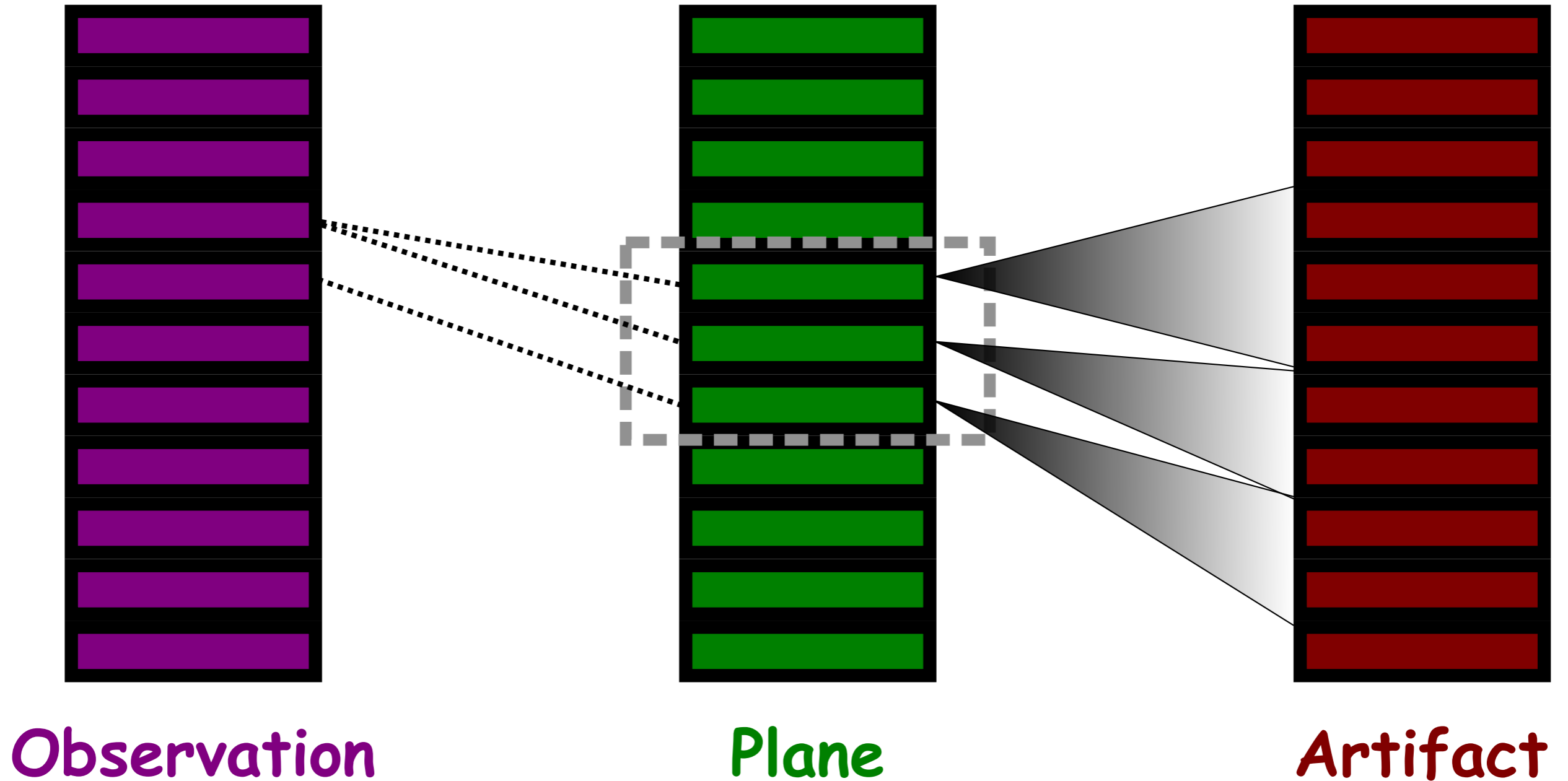
Fetching **Artifact** Results is Even Worse



Fixed by Making UUID's Spatial

- Replace the upper 32 bits of the ID columns with a 32 bit spatial index.
- Modified version of Q3C index
 - Hilbert Ordering
 - Less area variability via a quadratic remap
- The geometry for an **Observation** row comes from it's first **Plane** row.
- The geometry for an **Artifact** row comes from the **Plane** row that it belongs to.


Spatial Queries Are Now Fast



Fast For Some Other Uses As Well

- A search by a PI for their observation might start with looking for a particular **Observation** and then joining against **Plane** and **Artifact**.
- **Observation**'s usually look at one place in the sky, so fetching the **Plane** and **Artifact** rows will also be fast.
- A search for **all** of a PI's data would still have to skip around the **Observation** table.

Except When It Is Not

- Postgres can decide to do full table scans if you limit the number of records.....
- We do not have a good solution right now.

SIA2 is Easy

- After doing all of this work (which took many many months), implementing SIA2 is straightforward.
- This is no accident, since CAOM follows the ObsCore data model.
- Without that metadata regularization, SIA2 would be quite difficult to implement.
- This is in contrast to SCS, which places very few constraints on what columns are present.

SIA2 is Necessary to Test CAOM

- Asking astronomers to write joins to look at CAOM data is just going to result in a bad day for everyone.
- The parameter-based query allows them to quickly drill down to exactly what they want.

Spectra in SIA2?

- No?
 - Not in the name
 - The text for Data Product Type (dptype) says it will only be an image or cube
- Yes?
 - SIA2 services are **required** to support searching by resolving power.

SIA2 Makes SSA Obsolete?

- All of the mandated SSA search parameters and most of the recommended ones are included in SIA2.
- The data retrieval part of SSA is somewhat covered by SODA.

SSA is Also Easy

- If you are only trying to be 'query-compliant'.
- And the metadata is regularized.
 - Needed to disambiguate time cubes from spectral cubes by looking for whether resolving power is NULL.
- 'minimal-compliance' requires transforming our tables.
 - Maybe not hard, but more than an afternoon's work.

Time is Uncertain

- Projects sometimes only deliver observation timestamp and exposure time.
- Not always sure if that is the beginning, middle, end, or something else.
- In these cases, we make $t_{\min}=t_{\max}$.

Spectral Resolving Power

- Spitzer spectra capture multiple orders on a single CCD exposure.
- Different orders have different resolving power.
- Results are delivered as a single table.
- We store the minimum in the table.

Number of Points in a Spectra?

- SSA requires this as a result column.
- Resolving power and energy bounds seem like enough for queries
 - a similar argument can be made for the number of pixels in an image
- It is annoying for us to calculate, so we currently leave this NULL.