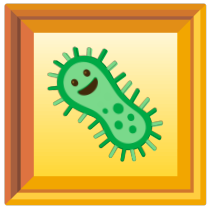
Image Data Cycles Brainstorm



Way Lab

9/26/2025

Ouroboros

Ancient Greek: "tail-eating"

Anonymous medieval illuminator; uploader Carlos Adanero
https://commons.wikimedia.org/wiki/File:Serpiente_alquimica.jpg

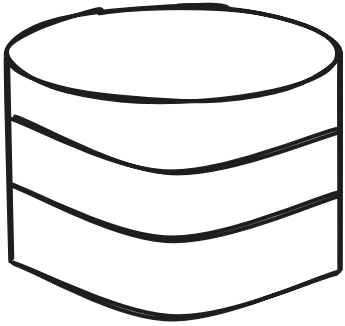
Armadillo girdled lizard (Ouroborus cataphractus)

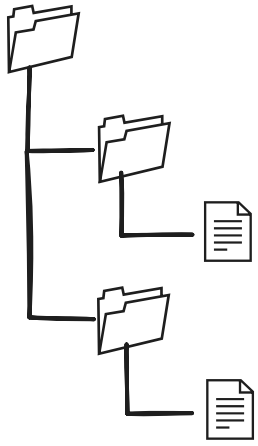
cndyntdhn
https://commons.wikimedia.org/wiki/File:Cu19.jpg

Most of the time we have this:

Profiles

Images





Invisible wall

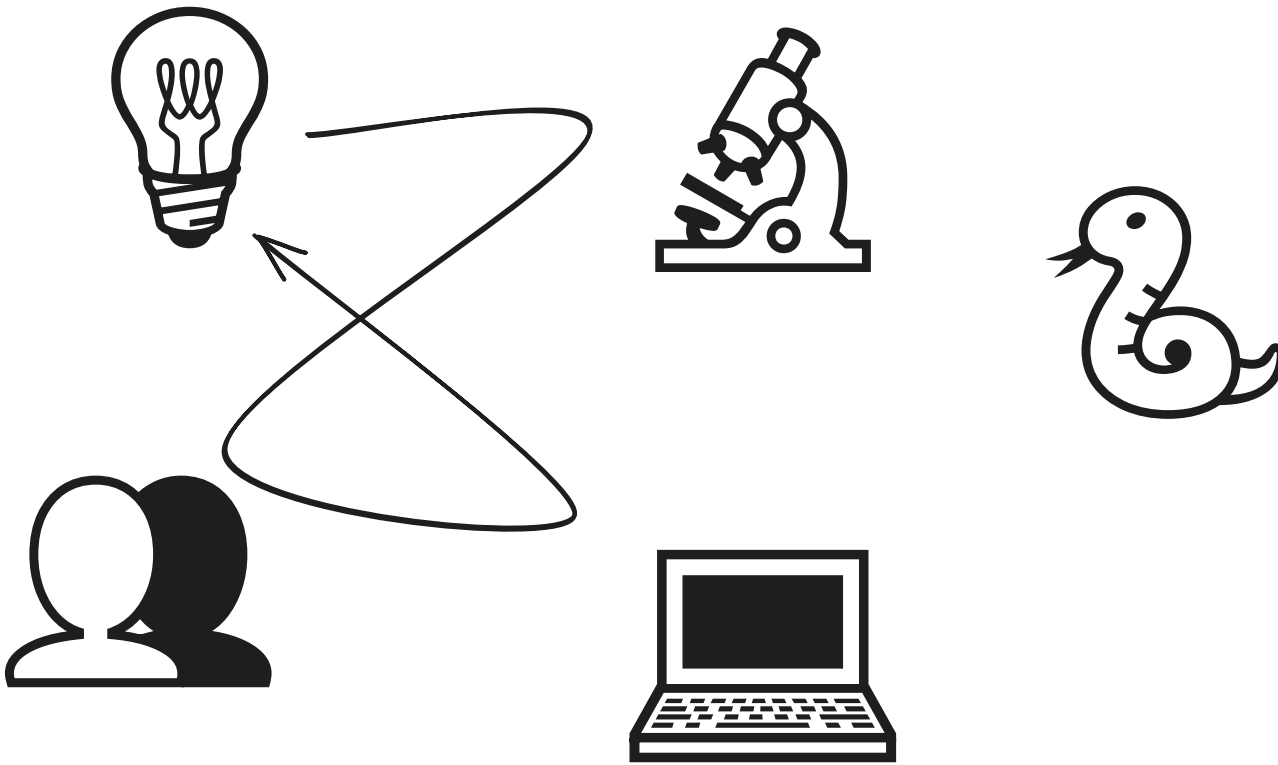
Databases

Files

This leads to a lot of friction:

- File formats
- Paths (schema)
- Data query duality
- Science isn't linear

Science isn't linear



Leaving images "behind" (linear)

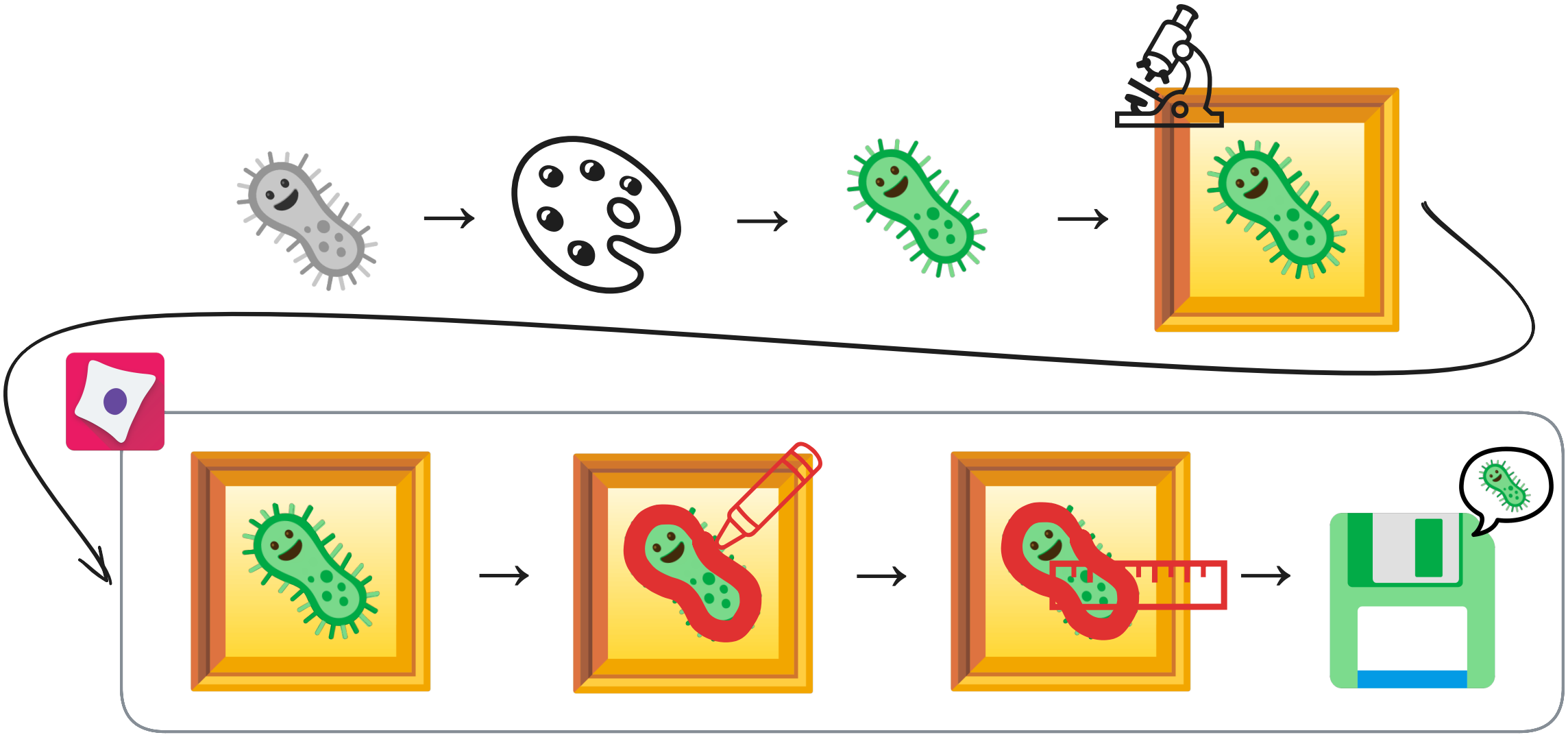
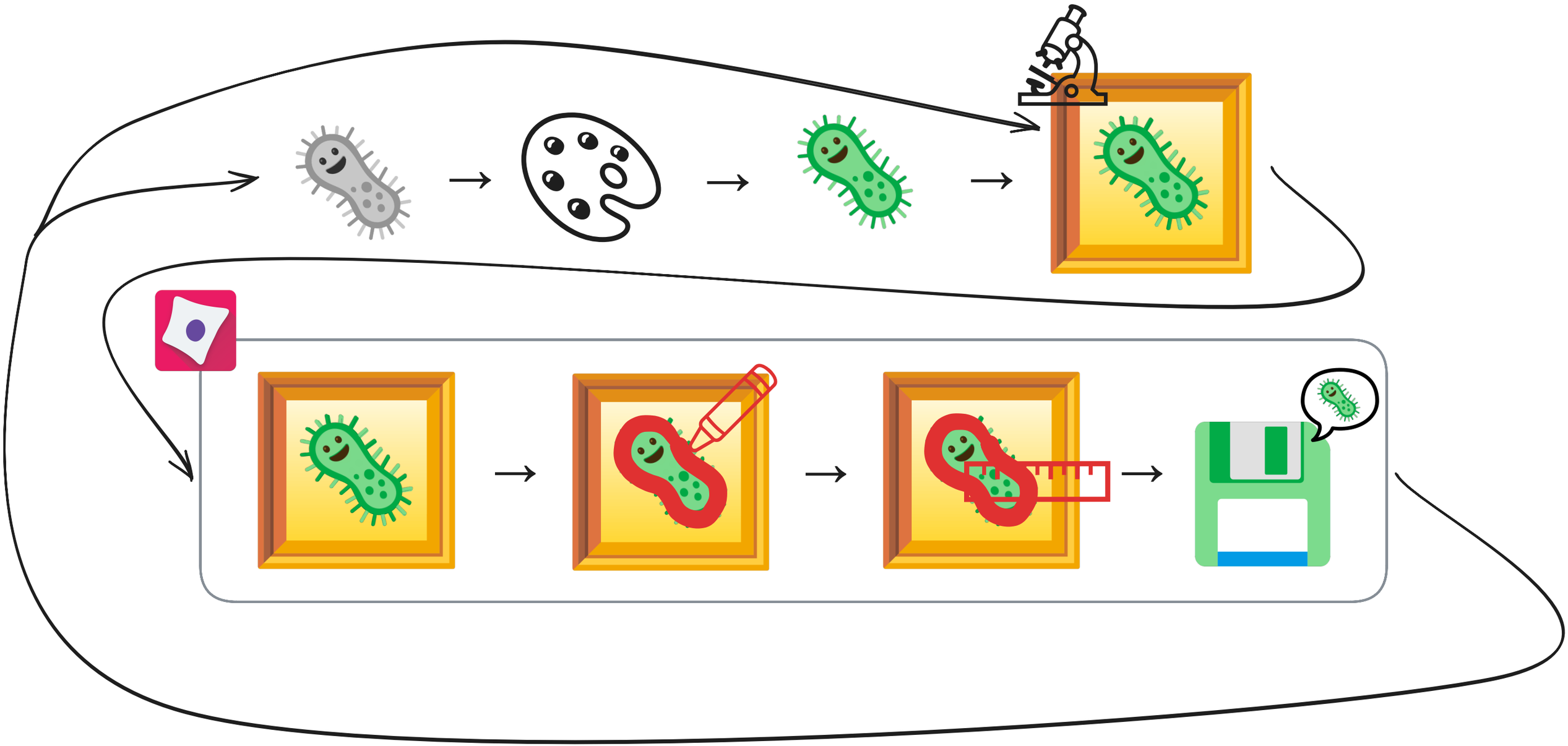
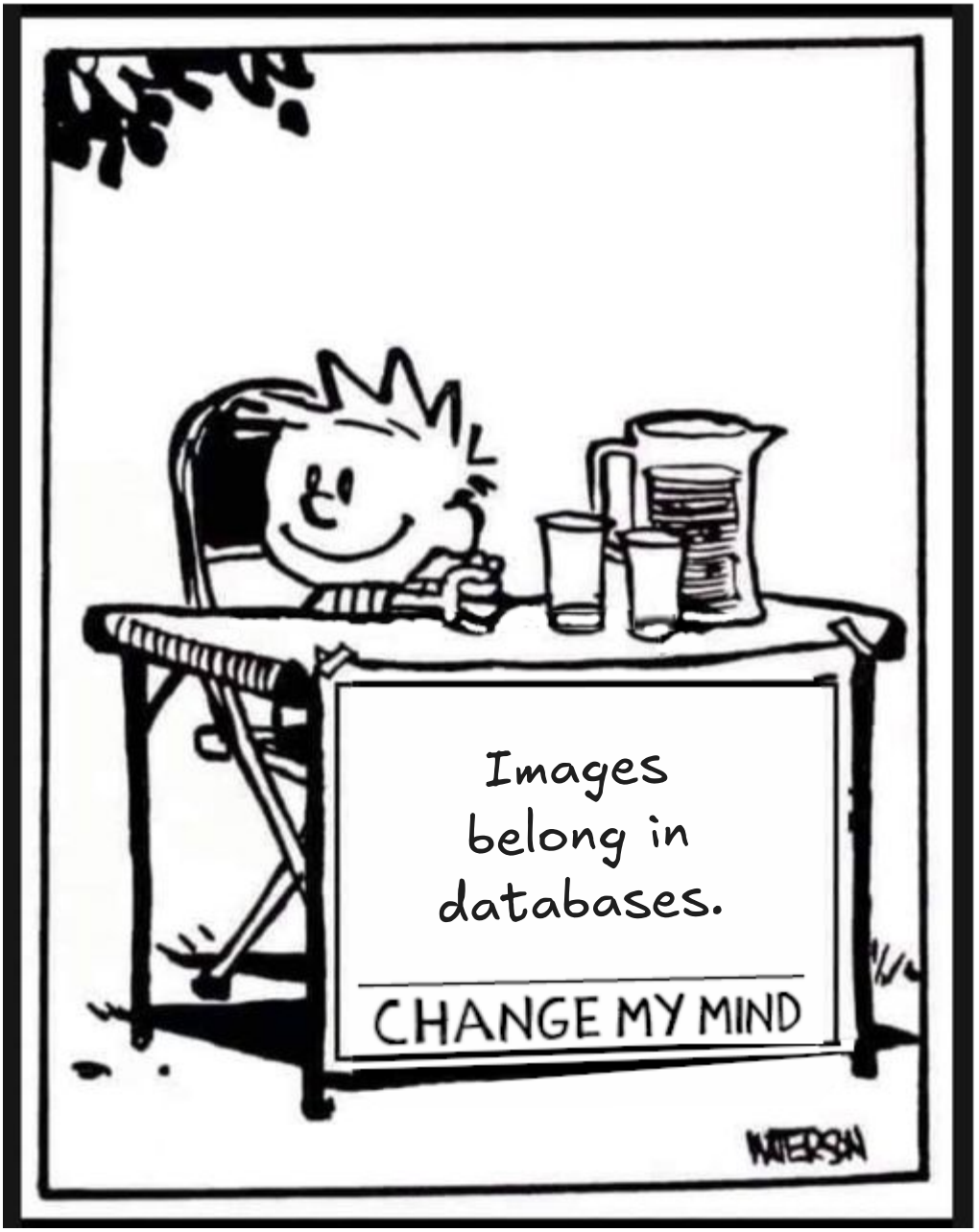
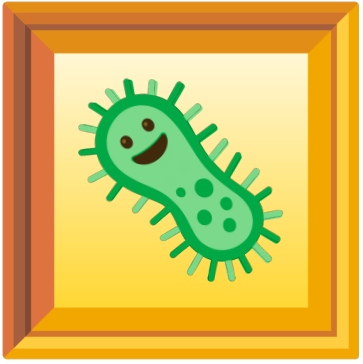


Image-based profiling cycles




Images
belong in
databases.
CHANGE MY MIND
WATTERSON

Images are data too!



Images

=

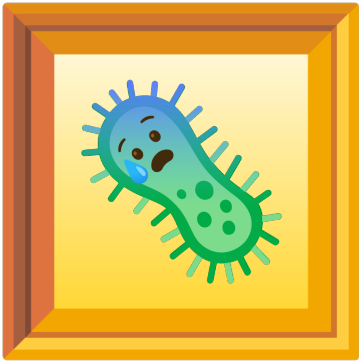
```
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
```

Pixels

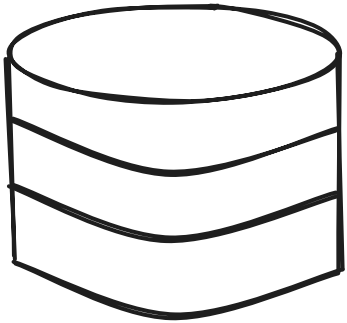
+

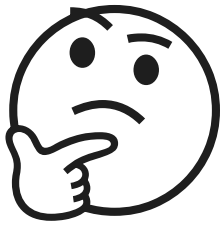
Metadata

Why are we always doing this?

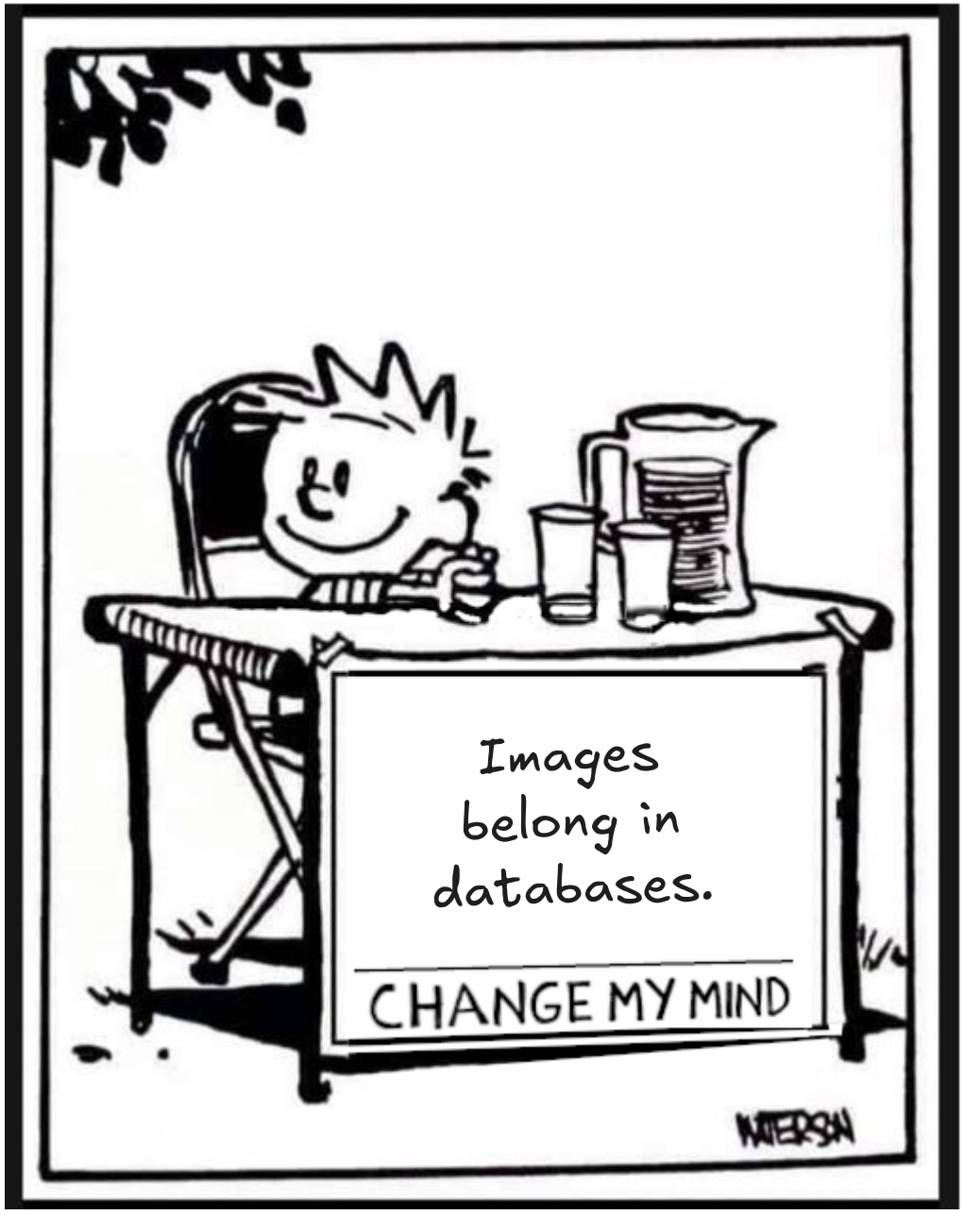


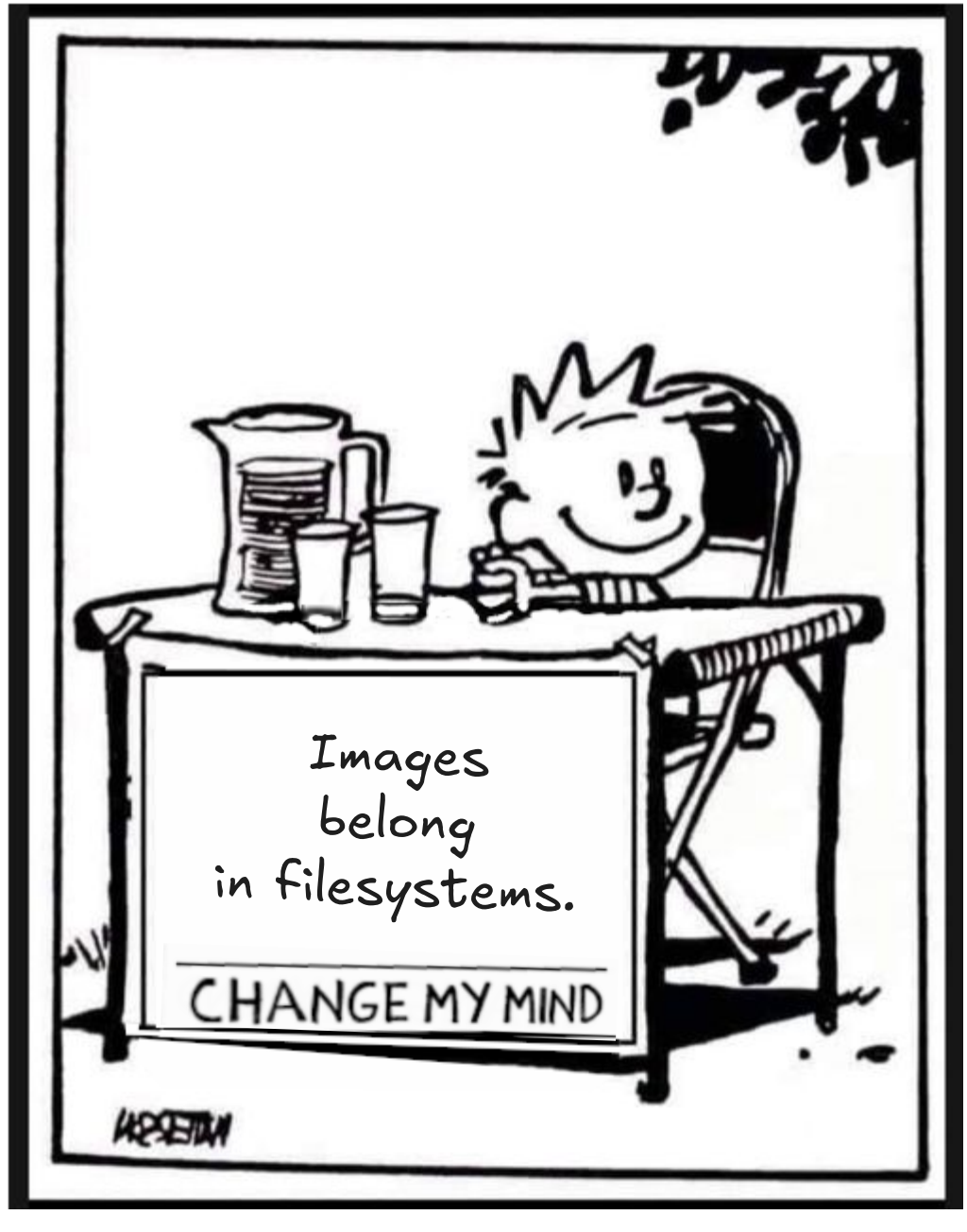
"Aren't we data too?!"



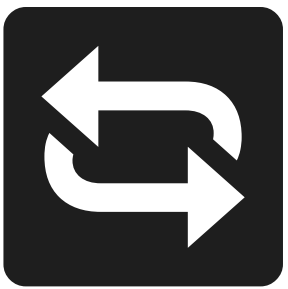


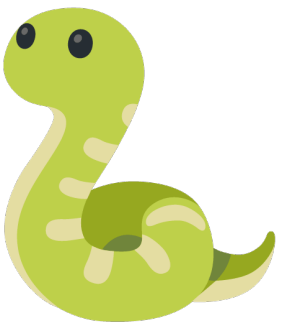
Hypothesis:
Storing image data in a database improves image-based scientific workflows by enabling faster, understandable, and more reliable retrieval and analysis of both pixels and metadata compared to file-based storage.


Images
belong in
databases.
CHANGE MY MIND
WATTERSON


Images
belong
in filesystems.
CHANGE MY MIND

History repeats itself!





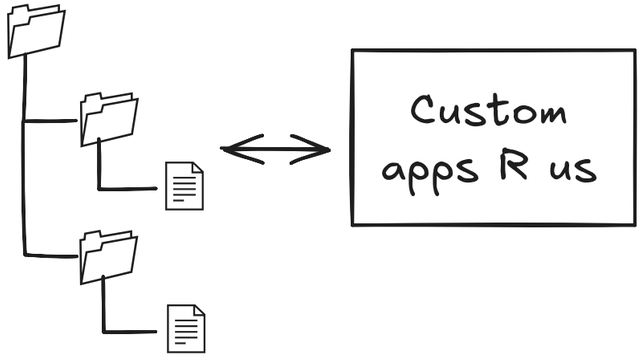
Where did databases even come from?

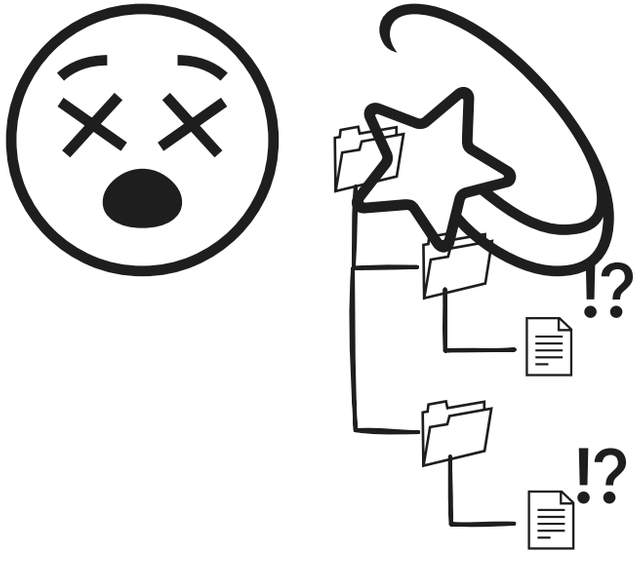
File-based systems (early days):

- Data stored in flat files or custom formats

- Applications managed indexing, relationships, and consistency

- Rigid and brittle—small schema changes broke programs





Why it wasn't enough:

- High complexity in application code

- Duplication of logic and data

- Poor scalability and data sharing across systems

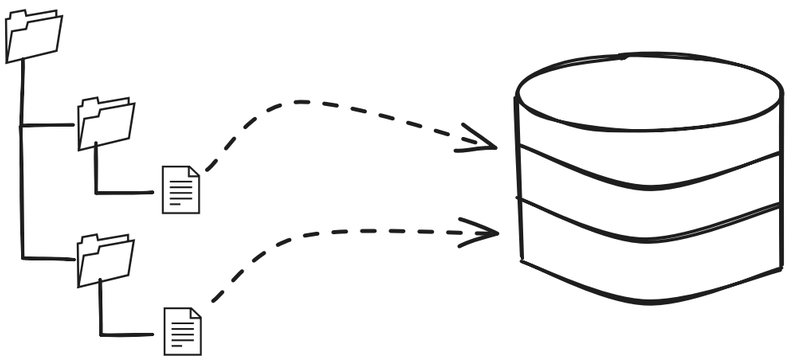
Relational Database Systems: (1970's)

- Abstracted away physical storage details

- Standardized with SQL for declarative queries

- Enabled data independence, integrity, and transactions

- Dramatically simplified development and maintenance

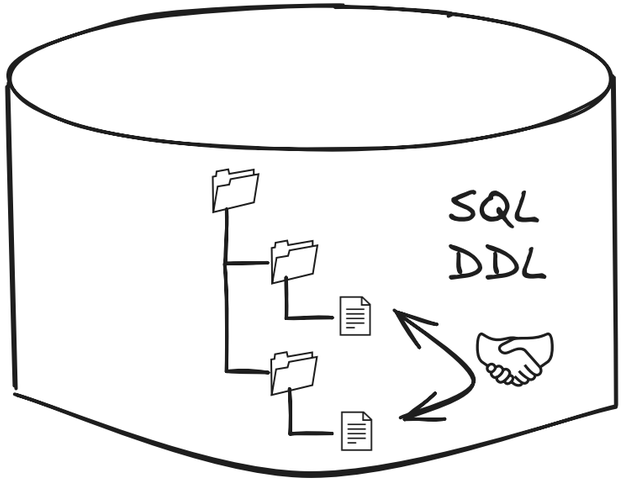


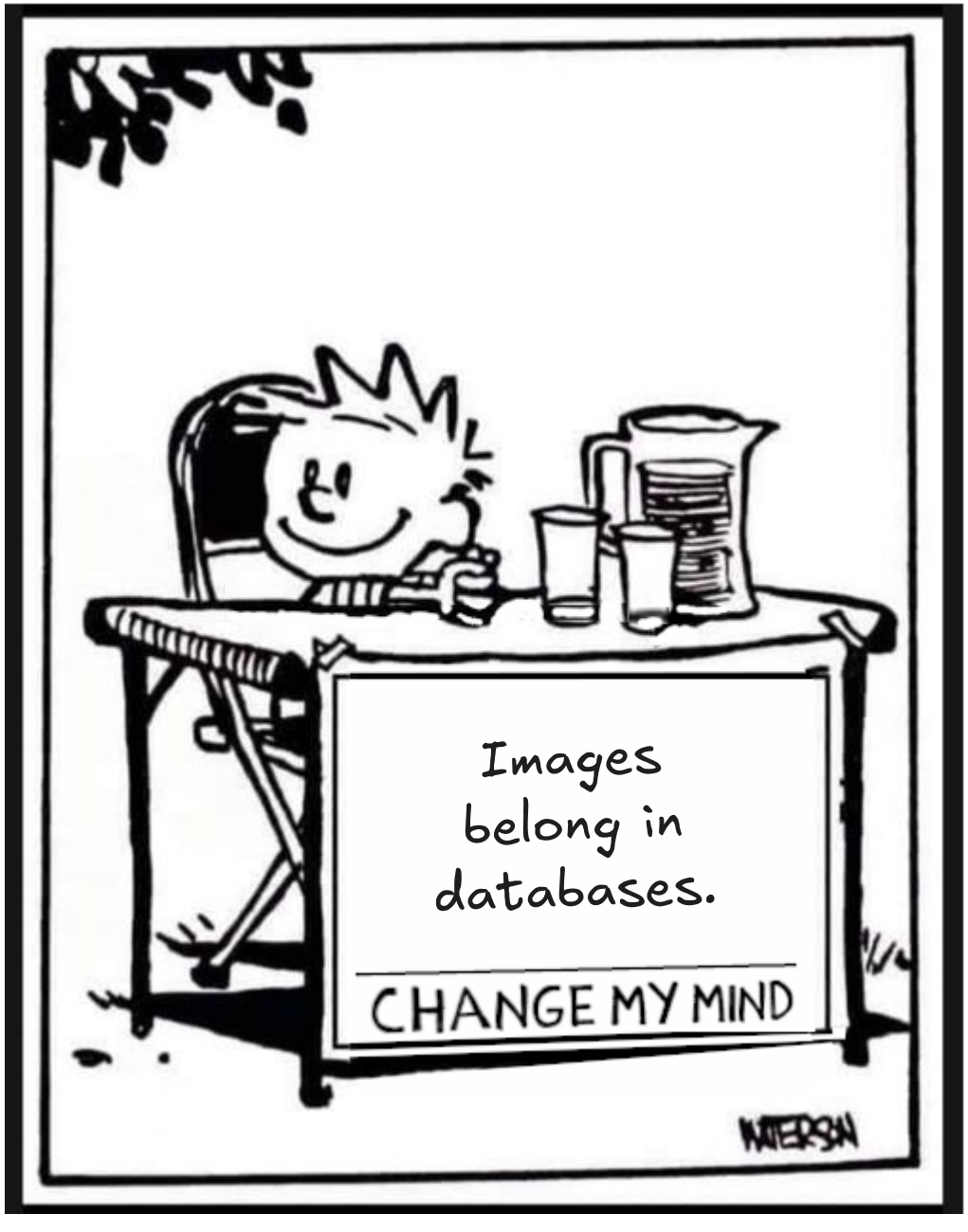
To be clear, with databases:

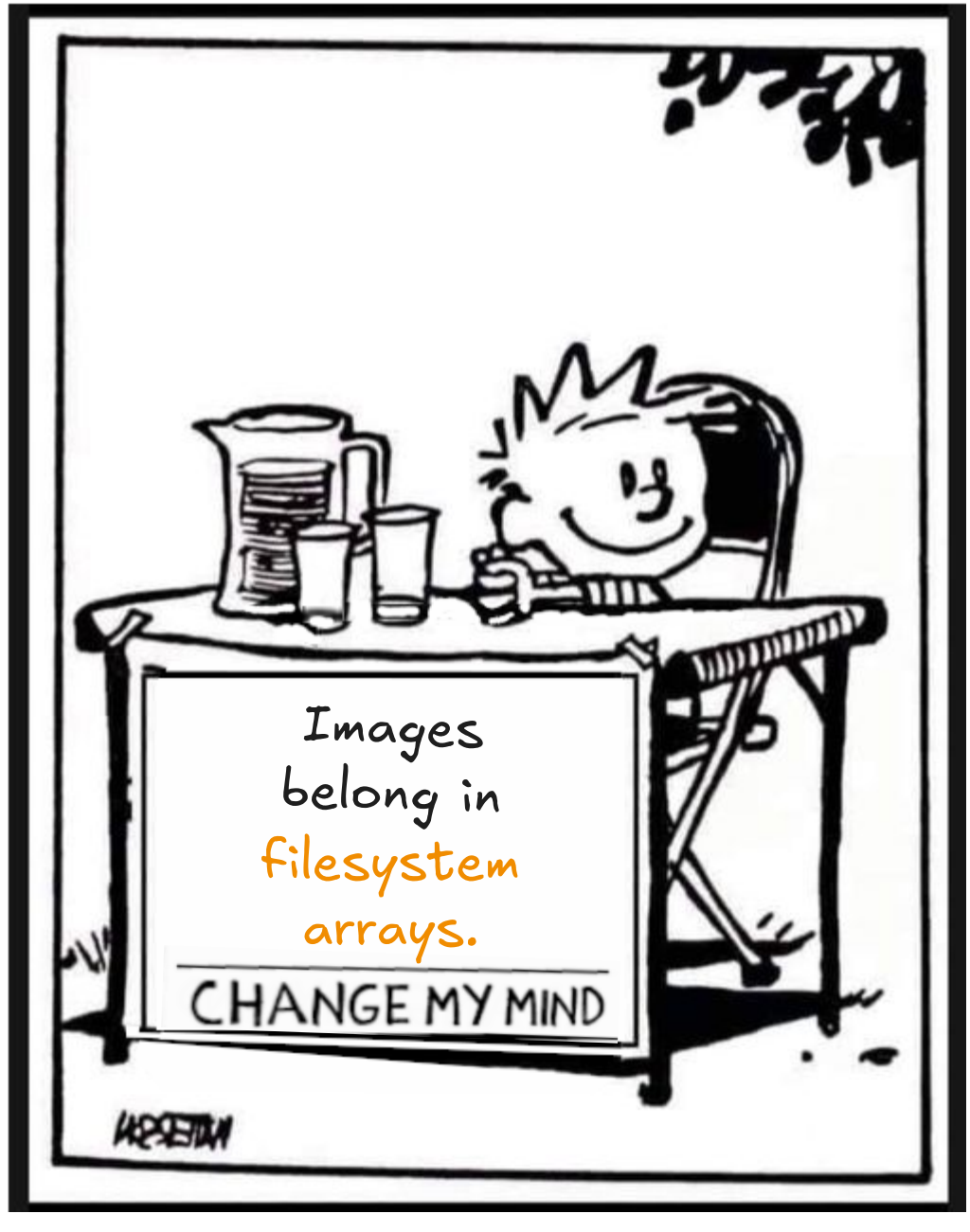
- We still have files, but they're consistently structured.

- We gained data querying grammar(!).

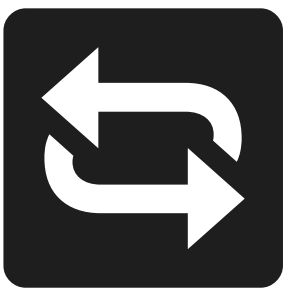
- We can since then define consistent relationships for data using for instance, data definition language (DDL).



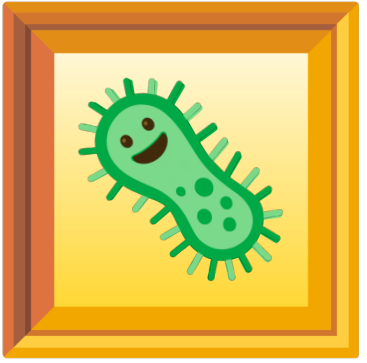

Images
belong in
databases.
CHANGE MY MIND
WATTERSON


Images
belong in
filesystem
arrays.
CHANGE MY MIND

History repeats itself!?



Remember: images are data too!



Images

=

```
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
```

Pixels

+

Metadata

Good data include strong types

1 = int32

2.3 = float64

"cowsays" = string

```
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
```

= ???

What even are all these?

```
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
```

!?

Dimensions?
Data types (e.g. float16, 32, 64)?
Fields / names?

Open Microscopy Environment (OME)
has figured these things out!

"OME is a consortium of universities, research labs, industry and developers producing open-source software and format standards for microscopy data."

OME Data Model

OME data model specification, code generator and implementation.

Links

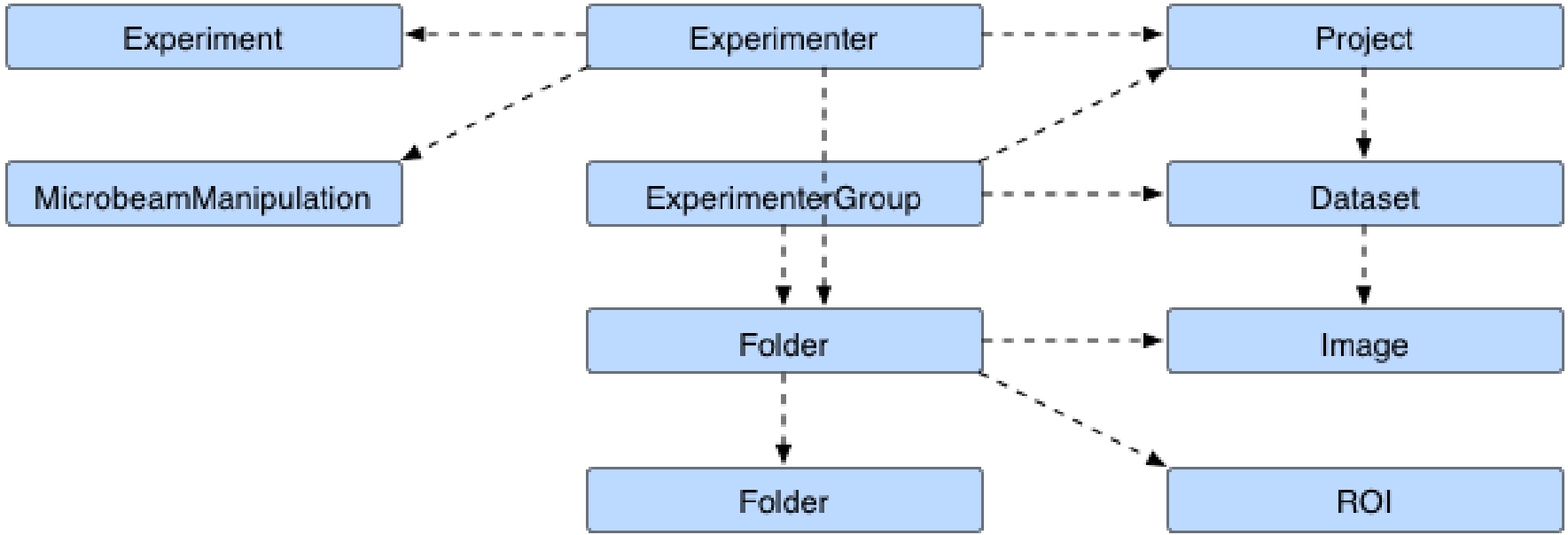
- Documentation
- Java API reference
- OME-TIFF sample images
- OME-XML sample images

https://github.com/ome/ome-model

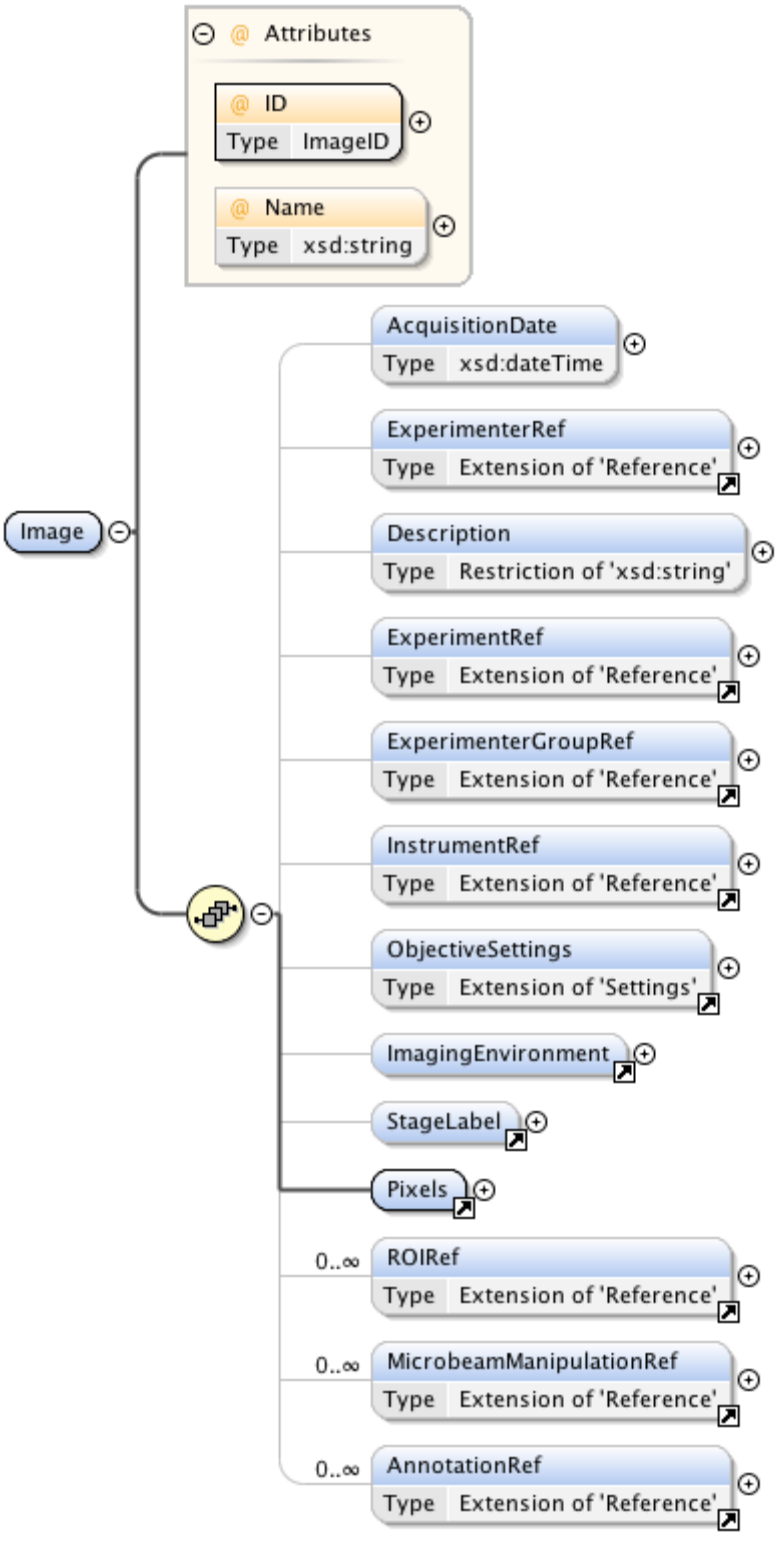
What even is a data model?

A data model is a structured way of describing
how data is organized, related, and stored.

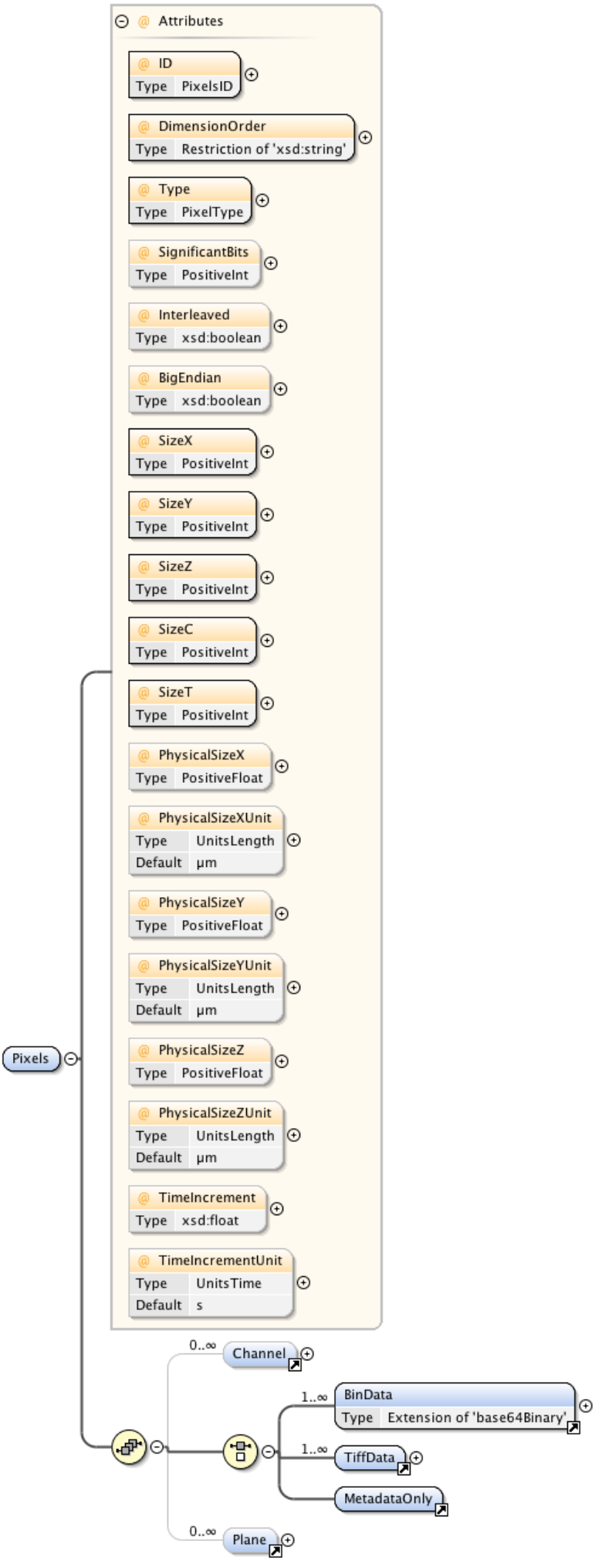
It defines the rules and shapes (like tables,
arrays, or objects) that give meaning to raw information.



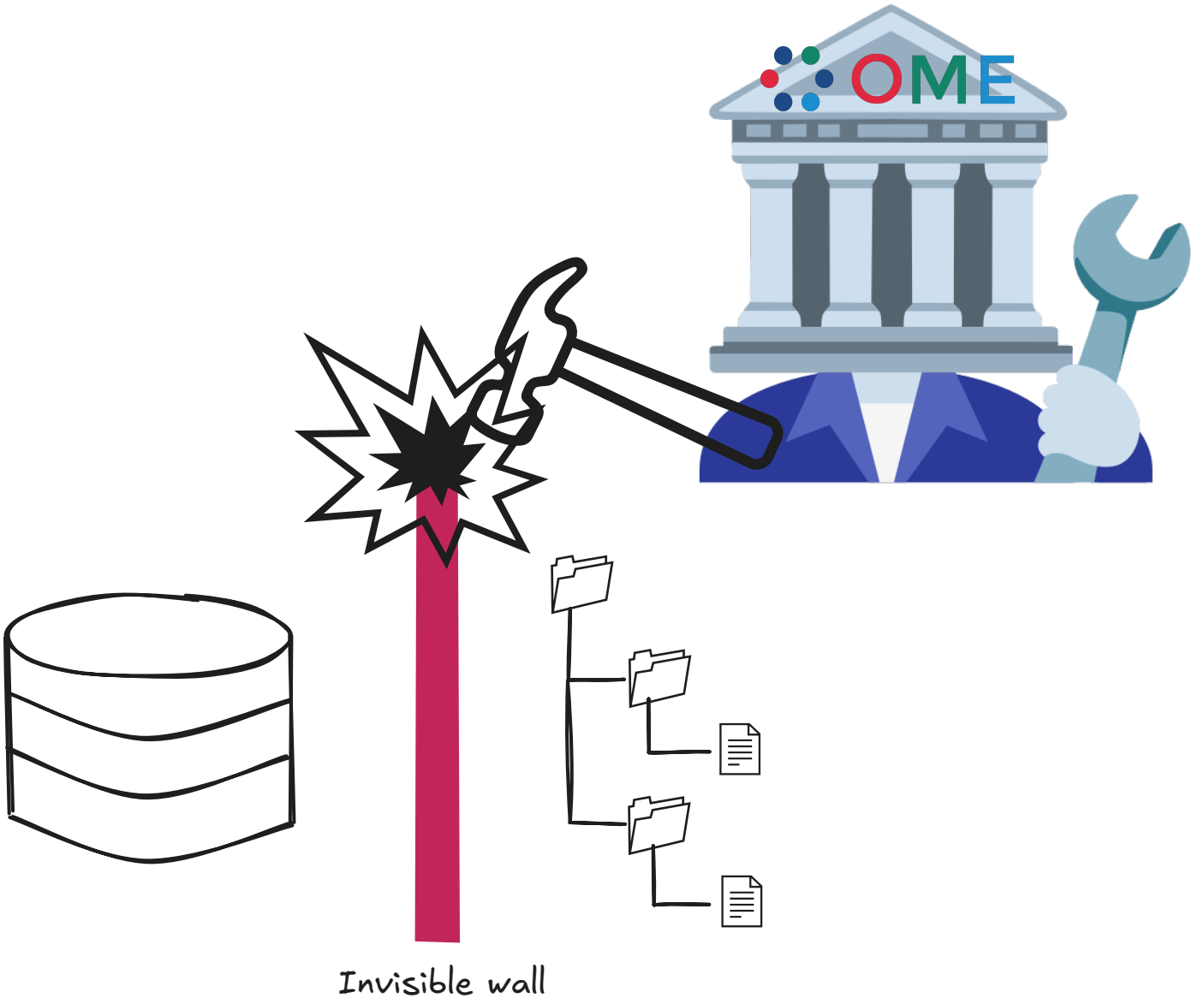
https://ome-model.readthedocs.io/en/stable/developers/model-overview.html



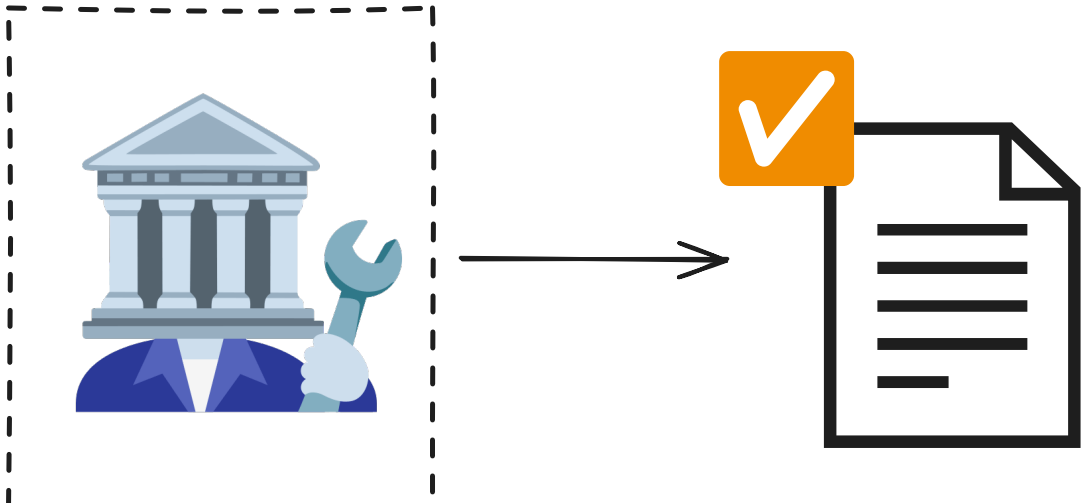
https://www.openmicroscopy.org/Schemas/Documentation/Generated/OME-2016-06/ome.html



https://www.openmicroscopy.org/Schemas/Documentation/Generated/OME-2016-06/ome.html

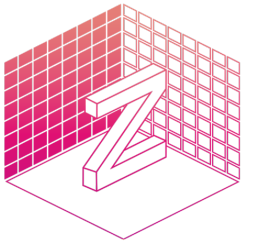

OME
Invisible wall

Data models still need to be applied somehow!



"Next generation file format"

OME-NGFF



old_data
- Z1EX_0999.F
  - 001_B.TIF

mydata.zarr
- array1
  - .zarray (describes array)
  - .zattrs (optional attributes)
  - 0 (array itself)

OME-TIFF

old_data

Z1EX_0999.F

001_B.TIF

new_data

Z1EX_0999.F

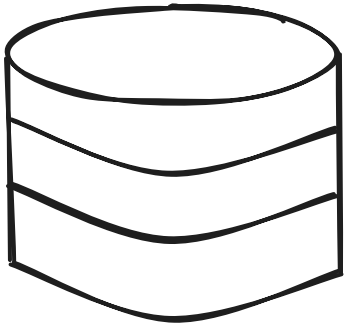
001_B.ome.tif

Structured image data, great!

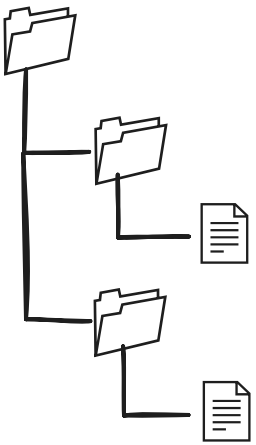
But what about:

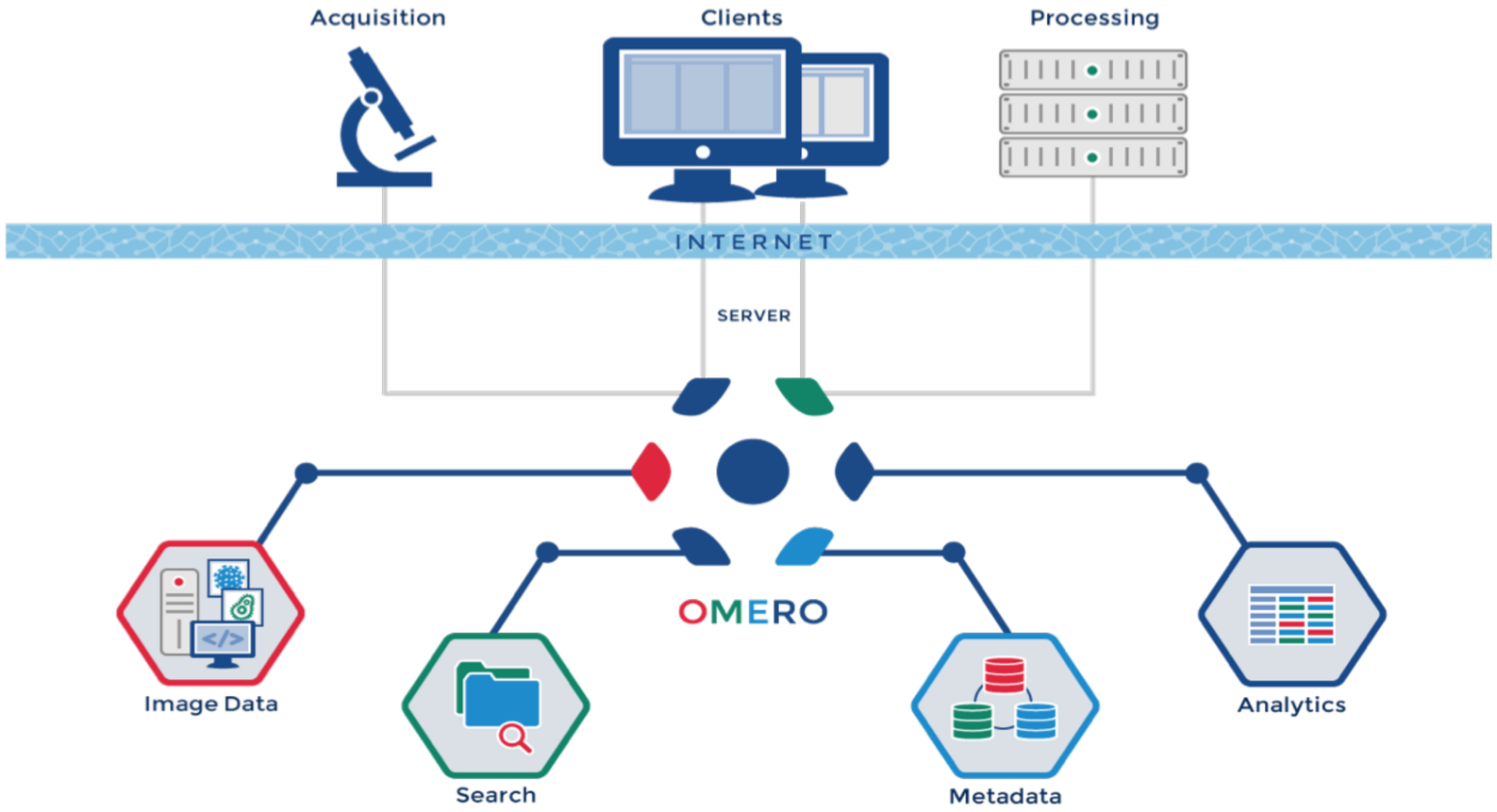
- Standard querying? (how do we extract data?)
- Relationships? (how do images relate to profiles?)
- Dramatic complexity reduction?

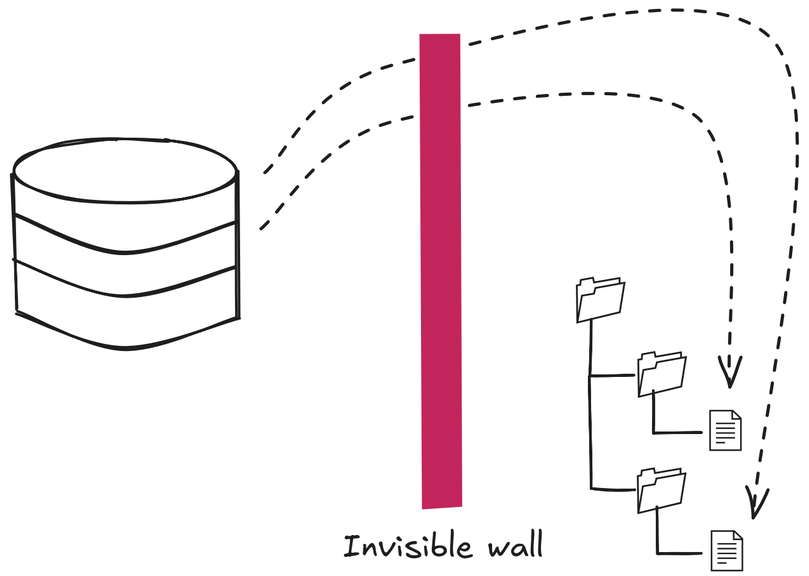
- We're still stuck behind the wall!



Invisible wall




Acquisition
Clients
Processing
INTERNET
SERVER
OMERO
Image Data
Search
Metadata
Analytics

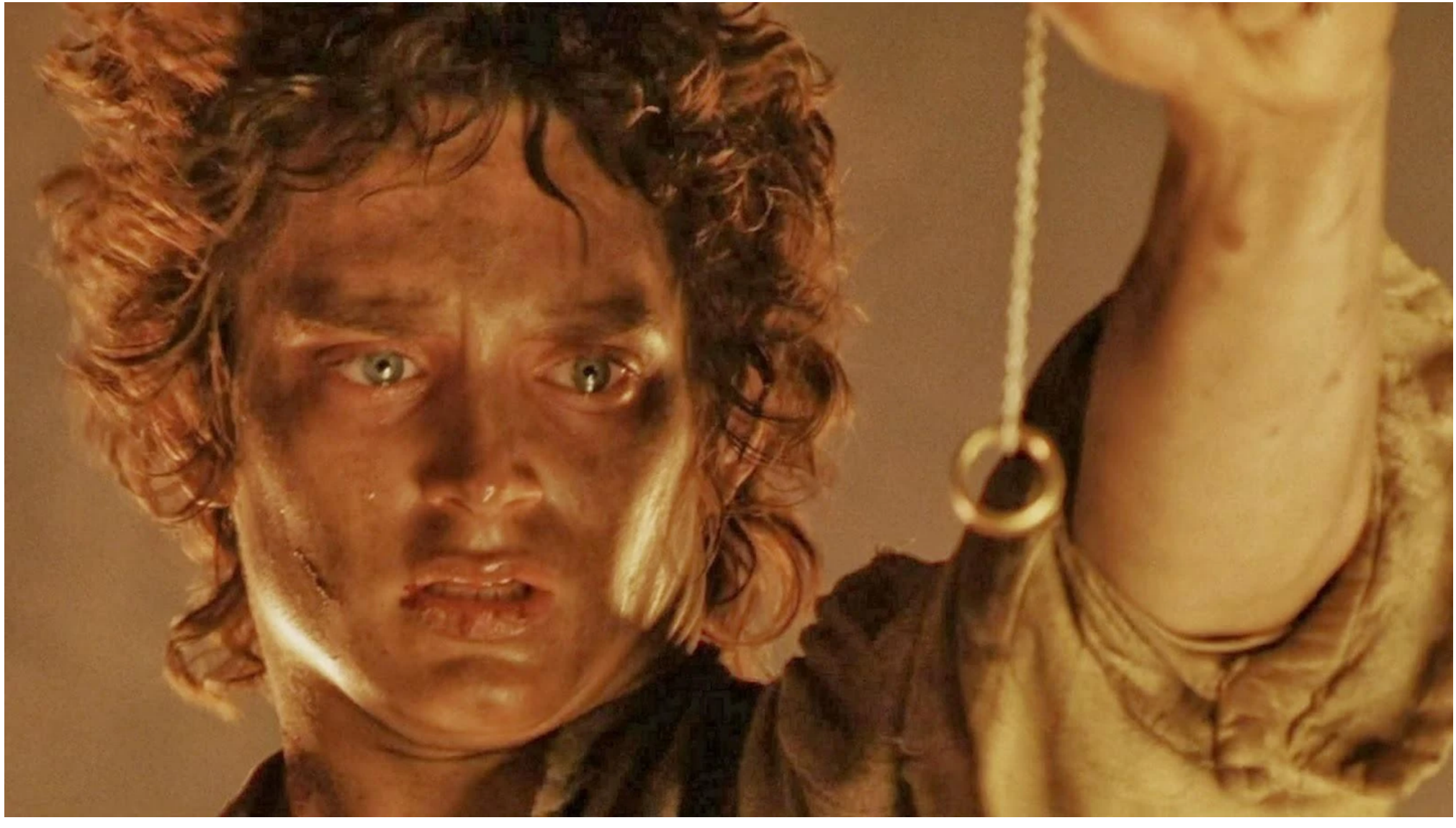

Invisible wall

OME and Omero documentation and publications appear to point to pixel data as "binary" and there are explicit design decisions to exclude that data from any database.

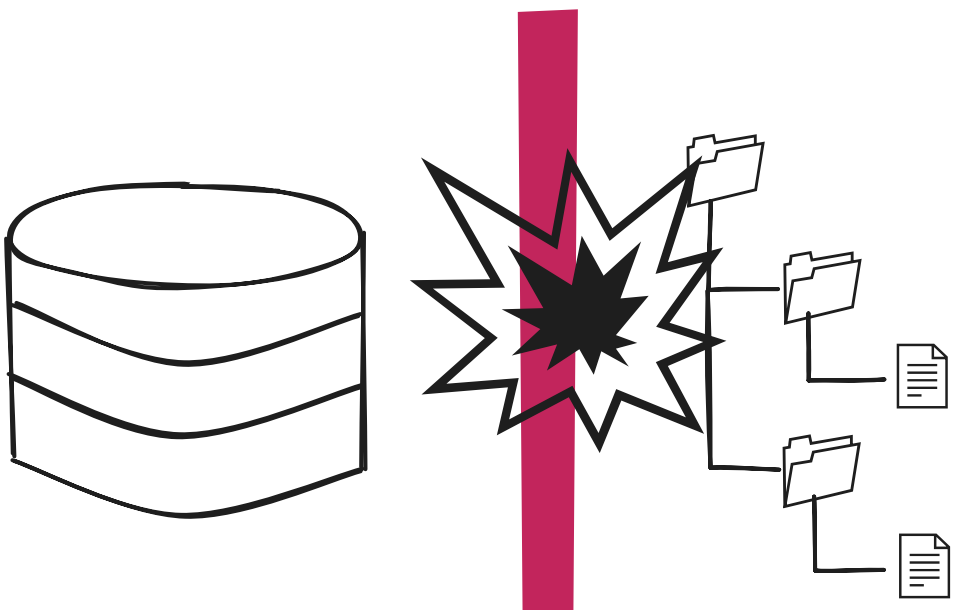
"Binaries" are often artifacts of structured data which require specific deserialization to process.

We do this with images all the time!

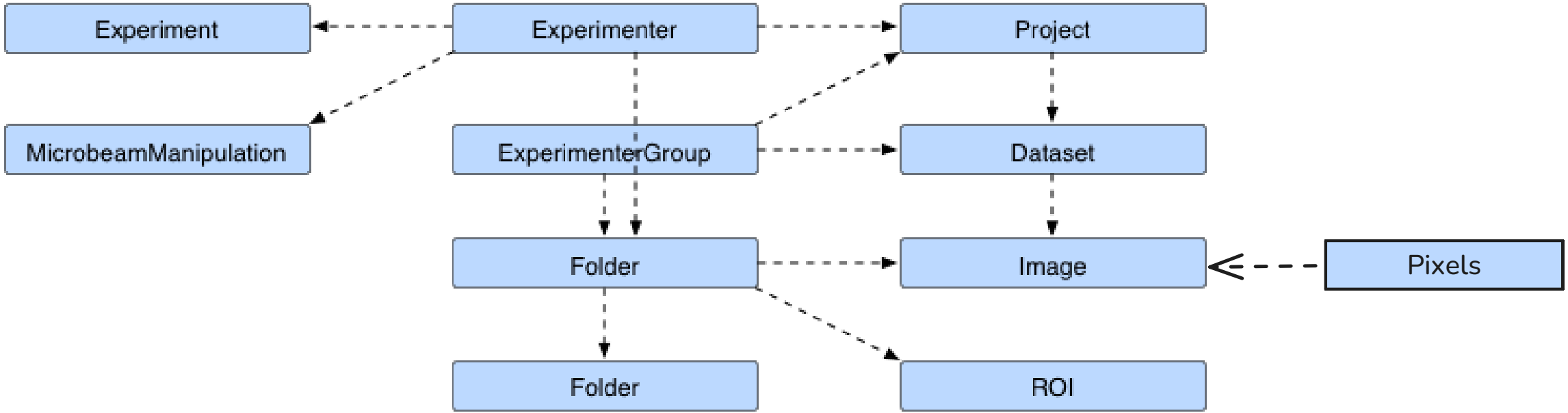
```
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
```



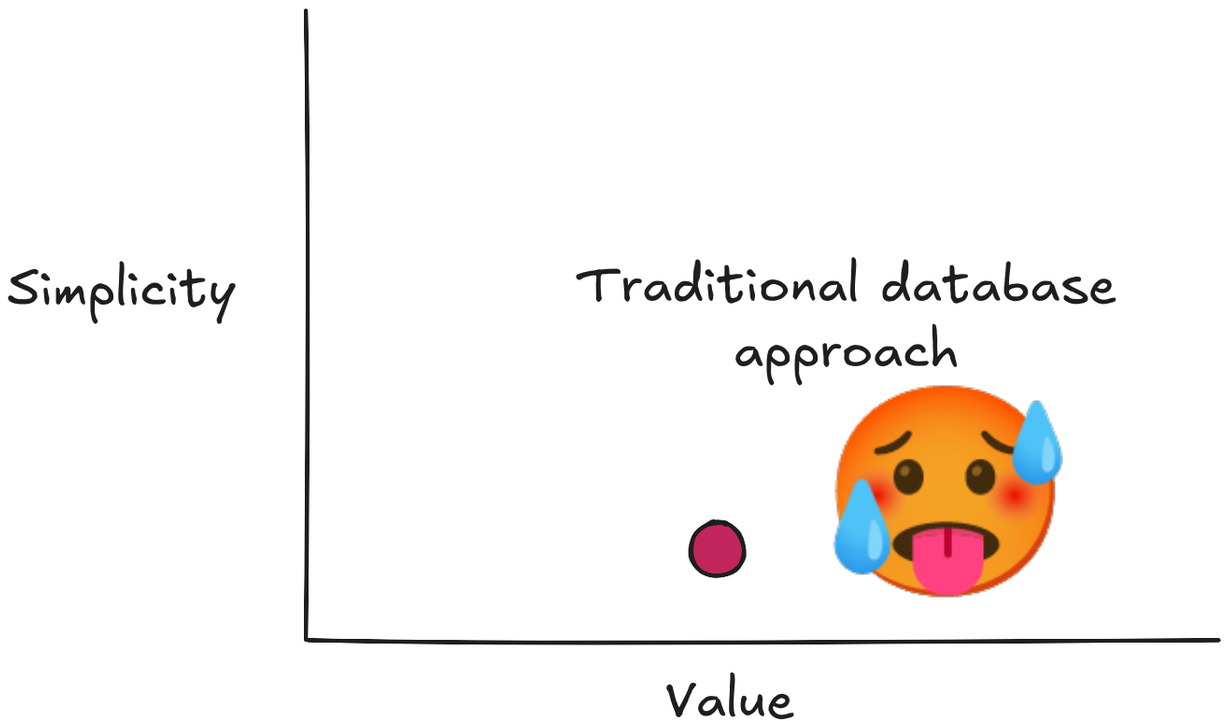
What could we do with a data model and format that doesn't have a wall between images and other information?



A more complete data model - traditional database approach

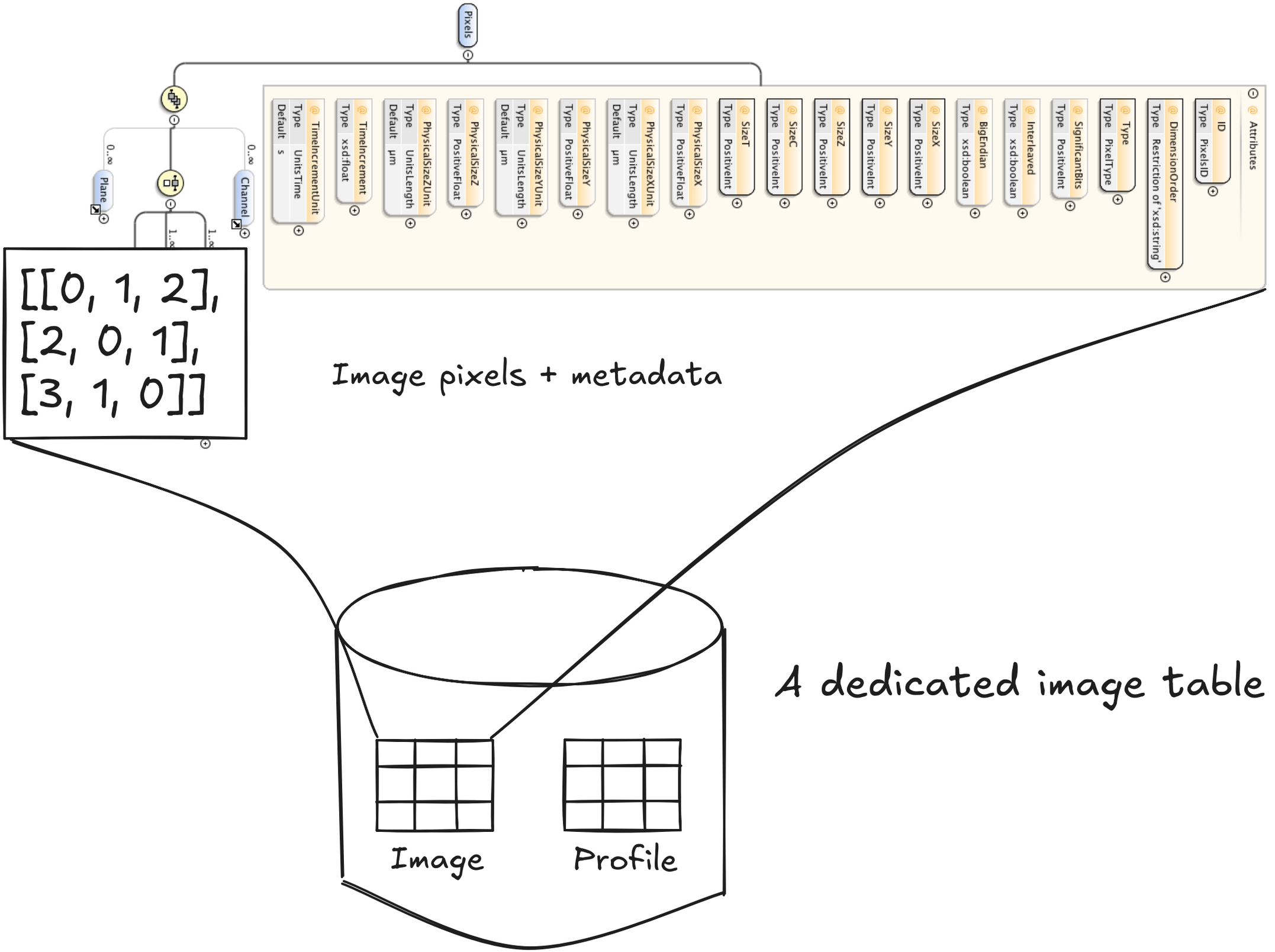


Perceived simplicity vs value (to implement and use)

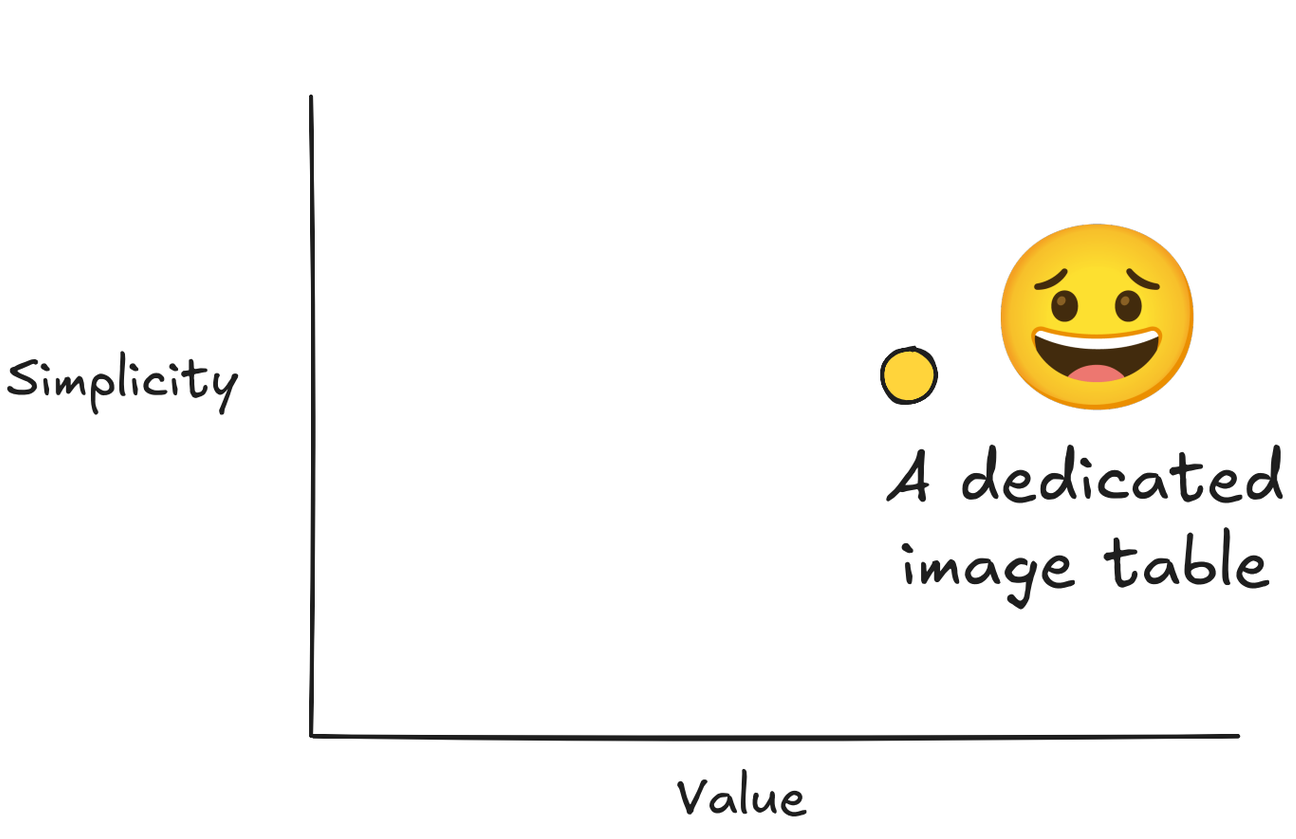


Users

Computer Scientists

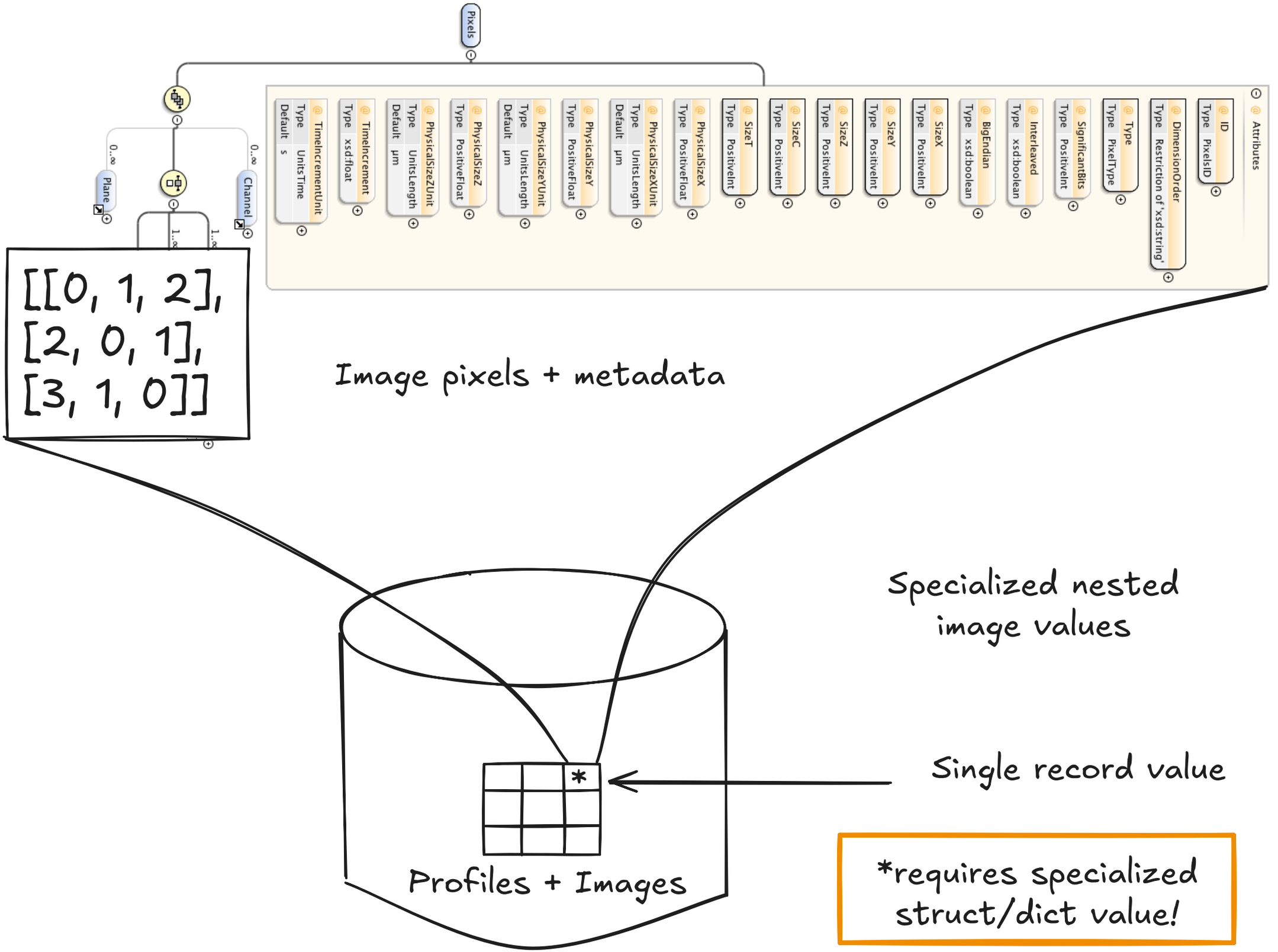

Pixels
Attributes
ID
Type PixelsID
DimensionOrder
Type Restriction of 'xsd:string'
Type
Type PixelType
SignificantBits
Type PositiveInt
Interleaved
Type xsd:boolean
BigEndian
Type xsd:boolean
SizeX
Type PositiveInt
SizeY
Type PositiveInt
SizeZ
Type PositiveInt
SizeC
Type PositiveInt
SizeT
Type PositiveInt
PhysicalSizeX
Type PositiveFloat
PhysicalSizeXUnit
Type UnitsLength
Default µm
PhysicalSizeY
Type PositiveFloat
PhysicalSizeYUnit
Type UnitsLength
Default µm
PhysicalSizeZ
Type PositiveFloat
PhysicalSizeZUnit
Type UnitsLength
Default µm
TimeIncrement
Type xsd:float
TimeIncrementUnit
Type UnitsTime
Default s
Plane
Channel
0..∞
1..∞
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
Image pixels + metadata
Image
Profile
A dedicated image table

Perceived simplicity vs value (to implement and use)

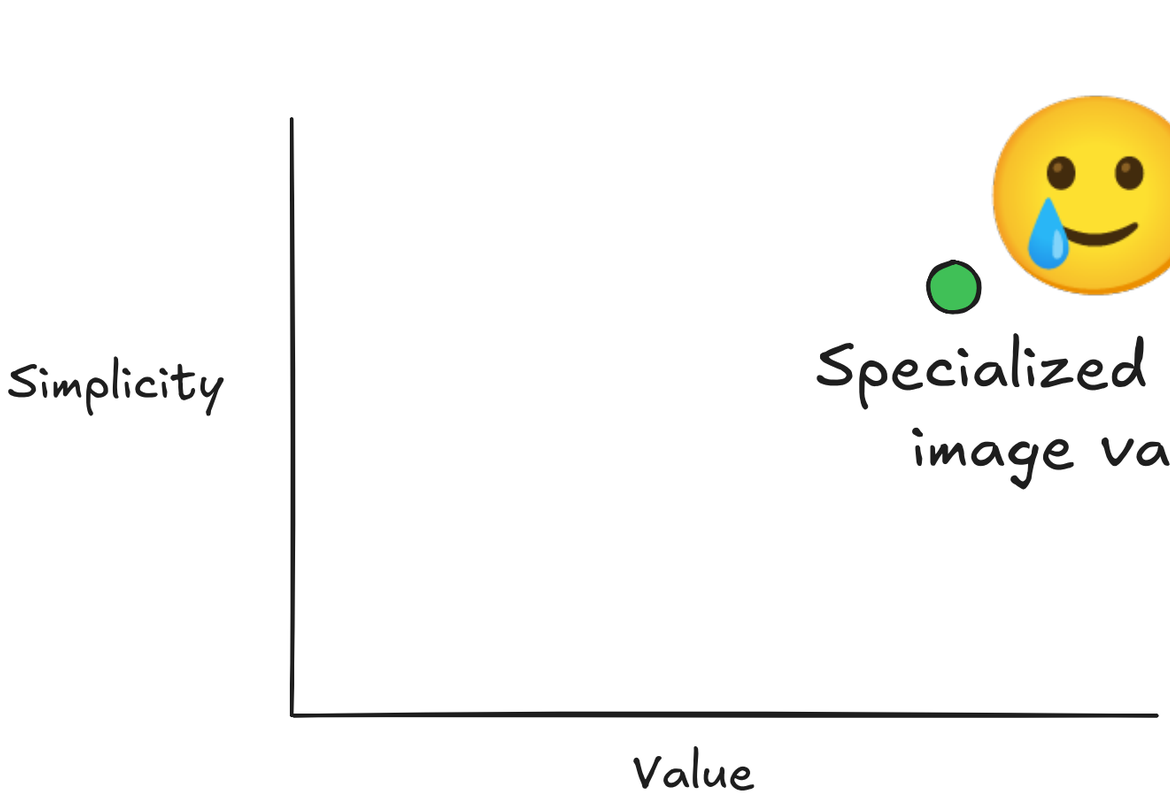


Users

Computer Scientists


Pixels
Attributes
ID
Type PixelsID
DimensionOrder
Type Restriction of 'xsd:string'
Type
Type PixelType
SignificantBits
Type PositiveInt
Interleaved
Type xsd:boolean
BigEndian
Type xsd:boolean
SizeX
Type PositiveInt
SizeY
Type PositiveInt
SizeZ
Type PositiveInt
SizeC
Type PositiveInt
SizeT
Type PositiveInt
PhysicalSizeXUnit
Type UnitsLength
Default µm
PhysicalSizeX
Type PositiveFloat
PhysicalSizeYUnit
Type UnitsLength
Default µm
PhysicalSizeY
Type PositiveFloat
PhysicalSizeZUnit
Type UnitsLength
Default µm
PhysicalSizeZ
Type PositiveFloat
TimeIncrementUnit
Type UnitsTime
Default s
TimeIncrement
Type xsd:float
Plane
Channel
0..∞
1..∞
[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]
Image pixels + metadata
Specialized nested
image values
*
Single record value
Profiles + Images
*requires specialized
struct/dict value!

Perceived simplicty vs value (to implement and use)

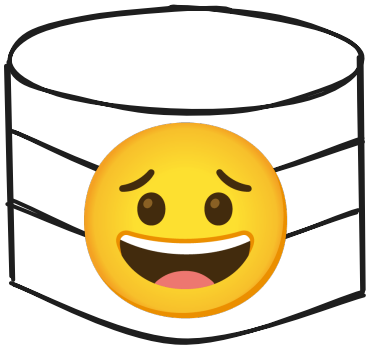


Specialized nested image values

Users

Computer scientists

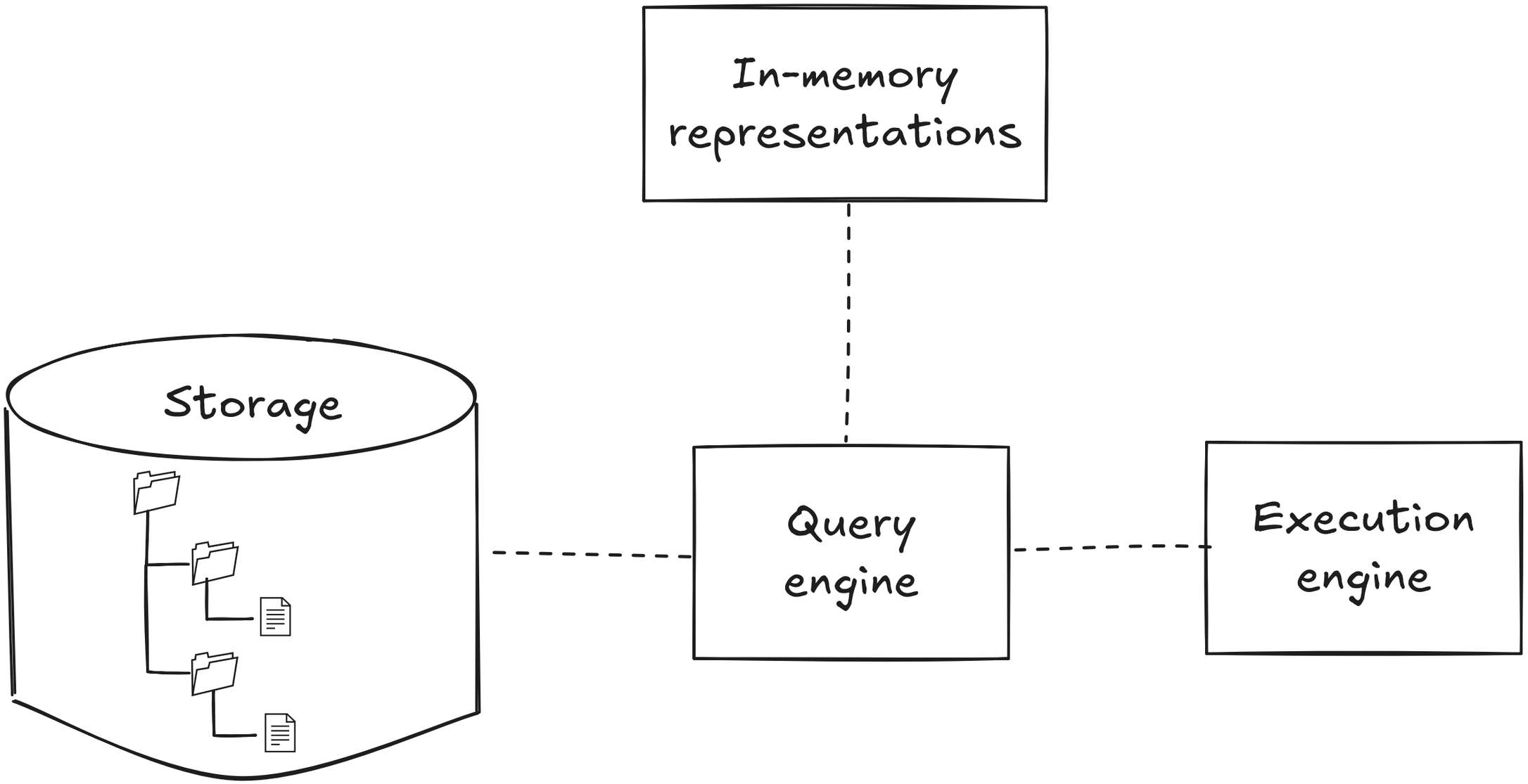
Support for multi-arrays within databases historically has not been great!



?

[[0, 1, 2],
[2, 0, 1],
[3, 1, 0]]

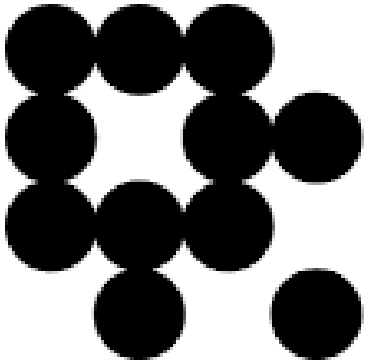
Composable, distributed databases



The Composable Data Management System Manifesto
https://doi.org/10.14778/3603581.3603604

Parquet is a distributed data table format which can be used in "deconstructed databases.".

Awkward array allows us to store "jagged" or "awkward" arrays within Parquet using custom data typing. Effectively your Parquet become Awkward Parquet - a format with many differences from "normal" Parquet.



LanceDB

Lance was created to handle AI/ML workloads efficiently. It builds on concepts from Parquet and enables multi-array value support over distributed data storage.

Some existing progress towards image pixels in values

serpula_rasa / src / serpula_rasa / example.ipynb

Preview | Code | Blame 1170 lines (1170 loc) · 211 KB

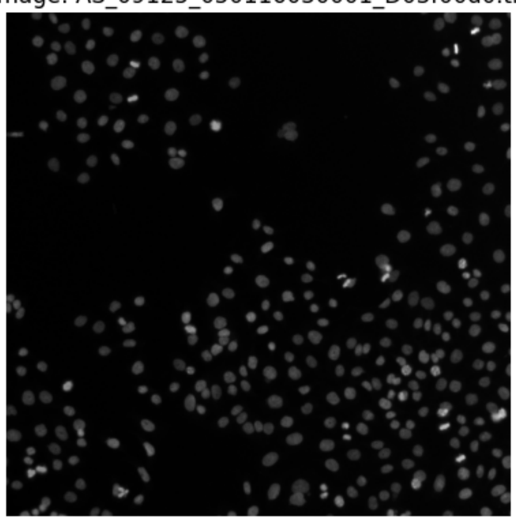
In [8]:
```python
# show the images table
db.open_table("masks").to_pandas()
```

Out[8]:

	filename	algo_name	algo_version	model_type	channels	n_objects	height	width	dtype	image
0	AS_09125_050116030001_D03f00d0.tif	cellpose	4.0.6	nuclei	(0, 0)	329	512	512	int32	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

In [9]:
```python
# Iterate through the records and display the images
for index, row in db.open_table("images").to_pandas().iterrows():
    img_array = np.array(row["image"], dtype=row["dtype"]).reshape(
        row["height"], row["width"], row["channels"]
    )
    plt.imshow(img_array.squeeze(), cmap="gray")
    plt.title(f"Image: {row['filename']}")
    plt.axis("off")
    plt.show()
```

Image: AS_09125_050116030001_D03f00d0.tif



https://github.com/WayScience/serpula_rasa/blob/main/src/serpula_rasa/example.ipynb

Thank you!

What's next?